9-1-2003

A Manufacturing Execution System using Siemens' PC Based Automation Technology

Prakash Gandhi

A Manufacturing Execution System using Siemens' PC Based Automation Technology

Prakash Amar Gandhi

M.S. (Computer Integrated Manufacturing)

Thesis submitted in partial fulfillment of the requirements for the Master of Science in the department of Manufacturing Engineering Technology in the College of Applied Science and Technology of the Rochester Institute of Technology

November 2003

COLLEGE OF APPLIED SCIENCE & TECHNOLOGY

ROCHESTER INSTITUTE OF TECHNOLOGY

ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

MASTER OF SCIENCE DEGREE THESIS

The M.S. Degree Thesis of Prakash Amar Gandhi has been examined and approved by the thesis committee as satisfactory for the Thesis requirement for the Master of Science Degree in Computer Integrated Manufacturing

---

Dr. Sudhakar R. Paidy
Dept. of Industrial and Systems Engineering
Kate Gleason College of Engineering

Prof. S. Manian Ramkumar
Dept. of Manufacturing and Engineering Technology
College of Applied Science and Technology

Permission granted

A Manufacturing Execution using Siemens' PC Based Automation Technology

I, Prakash Amar Gandhi, hereby grant the permission to the Wallace Library of the Rochester Institute of Technology to reproduce my thesis in whole or part. Any reproduction will not be for commercial use or profit.

Date: 11/11/2003 Signature of Author: Prakash Amar Gandhi

Dedication

To my parents and family,

because of whom I am what I am and where I am

Acknowledgement

Throughout the time I have been working in this applied research study, many people have assisted me. It is impossible to acknowledge them all. Nevertheless, I would like to specifically thank the following individuals:

Dr. Sudhakar Paidy, for all the time he dedicated to my research work and for the knowledge he shared with me throughout the course of this study. This thesis would not have been possible without his help.

Prof. S. Manian Ramkumar, for his continuous support and valuable comments on my work.

Mr. Dennis Wilk, Mr. Barry Hawley, and Mr. Peter Stansky of Siemens' Energy and Automation for their constant help and technical support.

To Adwait Palsule, fellow graduate research assistant, for his support, advice and comments throughout the course of this study.

To all my friends who helped me from the beginning of my thesis for giving valuable support and comments and those who helped me to prepare for the final presentation.

Table of Contents

List of Figures

Appendices

No.	Appendix
A	Read Data from FoxPro Database and Write to WinCC Tags
B	Write Data from WinCC tags to FoxPro database
C	Real Time Data capture from WinCC into FoxPro Database
D	Write Data from .SEQ and/or .SQX file to WinCC tags
E	Handles for Runtime Archive Functions
F	Format of a .SQX file

Abstract

The focus of any manufacturing operation is to establish better yields, reduced cycle times, increase quality, and handle dynamic demand/resource fluctuations. Over the past few years many manufacturing companies have implemented Enterprise Resource Planning (ERP) systems and they have proved themselves to be successful in achieving these goals. However, real-time data is required in order to portray an accurate account of the day-to-day and/or hourly product manufacturing operations. Retrieval of this real-time data is a challenging task. A Manufacturing Execution System (MES) is a real-time information system that improves the performance of the shop floor operations by linking business planning, order entry, material management, purchasing and accounting to the controls on the factory equipment.

Siemens' PC-based automation technology is an emerging technology that appears to provide a robust architecture for integrating all elements of the manufacturing environment. Applications that range from simple control to distributed control and full-fledged MES can be developed using Siemens' architecture.

The primary focus of this thesis is applied research to facilitate the development of a Manufacturing Execution System to control a flexible manufacturing system, CAMCELL, using Siemens' PC-based automation technology and Microsoft's database technology. CAMCELL contains two CNC machining centers, assembly robots, and a vision system, all of which are interlinked by a material handling system. The software architecture of the CAMCELL is based on NIST's five level hierarchy. Specifically, it contains functional modules for order entry, scheduling, and routing. In addition to these functional modules, there are various support modules. In this study, we have developed software architecture to achieve vertical integration of the process control layer, the MES layer and the ERP layer. Using Siemens' WinCC software, real-time process data was collected and integrated into an MES database. The study demonstrates how order information stored in a high-level database is converted into useful information for the control layer. The study also demonstrates the ability of WinCC and Visual FoxPro to update the production data into the MES database. Various Operator interface and database screens are proposed for CAMCELL.

Section 1. Introduction

The key to any production plant is the ability to capture real-time parameters and influence the business process. However, the volume of information and the speed requirements make it impossible for human interference. Nevertheless, over the past few years, manufacturing companies have implemented large Enterprise Resource Planning (ERP) systems, which have proven to be very effective. It has been a challenge, however, to retrieve critical data from the manufacturing shop floor. This data is required in order to portray an accurate account of the day-to-day and/or hourly product manufacturing operations. The solution is a Manufacturing Execution System (MES): a path that connects the shop floor to the top-floor operations.[1]

Several companies that offer control and human/machine interfaces (HMI) are providing total industry solutions that may include the following: control hardware and software, advanced control applications, material handling equipment, and Manufacturing Execution System (MES) software. For example, Siemens *SIMATIC® IT Production Suite* is the collection of components to design, manufacture and maintain Manufacturing Execution Systems. SIMATIC IT Framework is a collection of highly integrated components designed to integrate the systems within each factory, standardize production across the entire enterprise and keep manufacturing processes aligned with supply-chain activity. By linking and completely integrating the worlds of production and management with one standard software, the SIMATIC IT Framework creates effective communications and process synchronization and coordination throughout a plant or series of plants.[2] Camstar's leading Manufacturing Execution Systems (MES) for global enterprises monitor and synchronize manufacturing activities across globally distributed plants, and link them in real-time to the enterprise. These systems, based on Camstar's *InSite Suite*, track products and orders on the plant floor, collect transactions for reporting to back-office applications including ERP and Customer Relationship Management (CRM), and electronically dispatch these orders or product requirements to the shop floor personnel. The *InSite* tracking framework also provides key data services to the shop floor and they including real-time quality data checks, yield monitoring, automatic system actions, and lot traceability for improved quality and process.[3] CIMNET produces a configurable Manufacturing Execution System (MES) called *Factelligence*™, which manages manufacturing shop floor operations, and provides the critical link to fill the manufacturing gap between Enterprise Resource Planning (ERP) and the factory floor. The CIMNET product utilizes the latest Microsoft Web and Oracle database technology to electronically manage production schedules, documents, product quality, machine efficiency, material yields and labor time.[4]

Since MES projects involve extensive system integration services, manufacturers need to be careful in selecting a supplier. Companies selecting MES vendors who need to be assured that their prospective partner will be capable of an ongoing relationship, are available to solve problems and update the system.[5]

The Siemens' PC-based automation technology is an emerging technology that appears to provide robust architecture for integrating all elements of the manufacturing environment. Applications ranging from simple control to distributed control and full-fledged MES can be developed using Siemens' architecture.

The primary focus of this thesis is applied research leading to facilitate the development of a Manufacturing Execution System to control a flexible manufacturing system using Siemens' PC-based automation technology and Microsoft's database technology. CAMCELL is used as a test bed. It is a flexible manufacturing system containing two CNC machining centers, assembly robots, and a vision system, all of which are interlinked by a material handling system. The software architecture of the CAMCELL is based on NIST's five level hierarchy.[6] Specifically it contains functional modules for order entry, scheduling and routing. In addition to these functional modules, there are various support modules.

This study aims to develop software architecture and information flow in the CAMCELL to achieve vertical integration. The study also aims to evaluate the Siemens' Automation System in terms of its important features in the context of Systems Integration. Various process control system applications are available that can acquire large quantities of data from the manufacturing process into a database, thereby providing data for the office area or third-party systems. These process control applications are also capable of passing the information in a reverse direction. Data for production originating in the enterprise level are handed over to the automation level by these software's. Process control applications also provide HMIs for the operators to view the work recipes and generate reports for management to analyze the production. This study aims at studying the Siemens' Automation System through these perspectives.

In Section Two, the evolution of the manufacturing system and how Manufacturing Resource Planning (MRP) gave birth to Enterprise Resource Planning (ERP) systems will be discussed, including the functions of MES in an information system architecture. This section also describe the control layer of a factory.

In Section Three, Structured Query Language, which is a standard language for many relational databases, will be discussed. The section will also analyze the normalization concept and three normal forms in a database.

In Section Four, the client/server computing, which is one of the widely used forms of distributed computing, will be discussed. The study will discuss presentation logic, processing, and the storage component of client/server architectures. The three types of client/server architecture -- File Server, Database Server and Three-Tier architecture -- are also described. This section contains a brief description on requirements for successful implementation of client/server projects.

Section Five will discuss Microsoft's Open Database Connectivity (ODBC) interface that helps to access data from a variety of database management systems. It describes four components of the ODBC architecture viz., Application, Driver Manager, Driver and the Data source. The section will also over File and machine Data Source Name (DSN). The section briefly describes Remote views in Visual FoxPro that facilitate the external data through ODBC. Finally it, describes ActiveX Data Objects (ADO) and the steps involved in connecting to a database through scripts.

Section Six will discuss one of the important capabilities of WinCC, that is, capturing real-time data from the field device into a MES database. This section elaborates on User Archiving and Global Script editors of WinCC. The Functions and Actions features of WinCC will also be discussed. The section briefly

describes the Open architecture of WinCC, which makes it capable of exchanging information with the enterprise and factory levels of the MES database.

Section Seven considers the CAMCELL through four different perspectives viz. the Manufacturing and Material Handling, Product/Process flow, Computer hardware hierarchy and the Software Architecture and Information flow perspective. This section describes the proposed database architecture of a Manufacturing Execution System for the CAMCELL manufacturing cell.

Section Eight examines the data flow between ERP, MES and the Control layer of a CAMCELL automation system. It describes the actual implementation of Siemens' automation and Microsoft's database technology in the CAMCELL. The section probes the established the procedure to read and write to and from WinCC and VFP databases. It also describes the steps involved in capturing real-time data from CAMCELL into a MES database. Proposed WinCC interface for workstations are also developed in this section.

Section Nine targets, will discuss the proposed VFP database screens for various users in the system. The concluding section summarizes the findings of this applied research study.

References:

[1]Manufacturing Execution Services, Inc. (2002)

<www.mes-erp.com/MES_solution.htm>

[2]Siemens AG (2003), Industrial Solutions and Services (I&S). (7 July 2003)

<www.is.siemens.de/itps/en/index.htm>

[3]Camstar Inc. (2003)

<www.camstar.com/products/insite.asp>

[4]Cimnetinc Inc. (2002)

<www.cimnetinc.com/>

[5]Intelligent Manufacturing. (1997) - Lionheart Publishing Inc
<www.lionhrtpub.com/IM/IMsubs/IM-2-97/ControlSuppliersMigrate.html>

[6]Dr. Paidy. S.R., & Dr. Reeve, Richard, "Software Architecture for a Cell Controller".

Section 2. Software Systems for Manufacturing

2.1 Enterprise Resource Planning

Since the Nineties, ERP systems have gained an explosive popularity among manufacturing enterprises worldwide.[6] An ERP (Enterprise Resource Planning) system is an integrated information processing system supporting various business processes, such as finance, distribution, human resources, and manufacturing.

Traditionally, companies developed separate computer applications to satisfy the needs of each of their functional segments, such as accounting, purchasing, inventory and planning. Such systems grew as inconsistent islands of information; hence, their consolidation was not possible when deemed necessary. As a result, decision-makers were denied access to timely information for making urgent business decisions. This gave rise to the development of an integrated system to be known as ERP that would address the information requirements of corporate heads.

Material requirement planning (MRP) and manufacturing resource planning (MRP II) did provide a certain level of integration. However, they primarily addressed the requirements of a manufacturing set-up. On the other hand, ERP addressed the information requirements of the entire enterprise. Moreover, it was not restricted to the current requirements of an organization but provided an opportunity for continuously improving and refining business processes.

ERP vanquishes the old stand-alone computer systems in finance, HR, manufacturing and the warehouse, and replaces them with a single unified software program divided into software modules that roughly approximate the old stand-alone systems. Finance, manufacturing and the warehouse all still get their own software, except now the software is linked together so that someone in finance can look into the warehouse software to see if an order has been shipped.[8]

Manufacturers who implemented Enterprise resource planning are finding ways to extend the usefulness of ERP as a backbone of distributing corporate information across the enterprise.

For a manufacturing company employing traditional manufacturing facilities and conventional MIS (Management Information Systems), FMS and ERP Systems may be the two major areas of investment for a company to stay competitive.[7]

The amount of investment for a typical FMS is about $5-10 million (US), and a typical ERP system is about $3-5 million (US). Thus, for a company that has invested in both FMS and ERP, it is essential to

have a suitable Manufacturing Execution System (MES) that can provide an adequate interface between the two.[7]

Over the past few years, many manufacturing companies have implemented ERP (Enterprise Resource Planning) systems and have proved themselves successful. However, it has been challenging to retrieve real-time data from the shop floor. This data is required to keep an account of day-to-day and/or hourly product manufacturing operations.

2.2 Manufacturing Execution System

MESA international defines MES as "Systems that deliver information enabling the optimization of production activities from order launch to finished goods. Using current and accurate real-time data, MES guides, responds to, and reports on plant activities as they occur. The resulting rapid response to changing conditions, coupled with a focus on reducing non-value added activities, drives effective plant operation and processes."[1]

An MES (Manufacturing Execution System) enables an enterprise to use standard software across multiple plants to optimize production process and connect the plants into enterprise supply chains. MES provides an enterprise wide, real time view of the complete manufacturing environment. It closes the gap between the ERP and production worlds for optimal results.[10]

The classical model of Computer Integrated Manufacturing (CIM) is divided into three layers: Planning, Execution and Control. The Planning layer includes ERP, Supply Chain Management (SCM) and Advanced Planning and Scheduling (APS) systems.[9] The control layer includes all real time shop floor control systems such as PLC, SCADA, DCS, CNC/DNC machines etc. MES occupies the middle execution layer and knits the manufacturing process together into an integrated whole, closing the gap between event-driven production systems and transaction-based planning and management systems.[9]

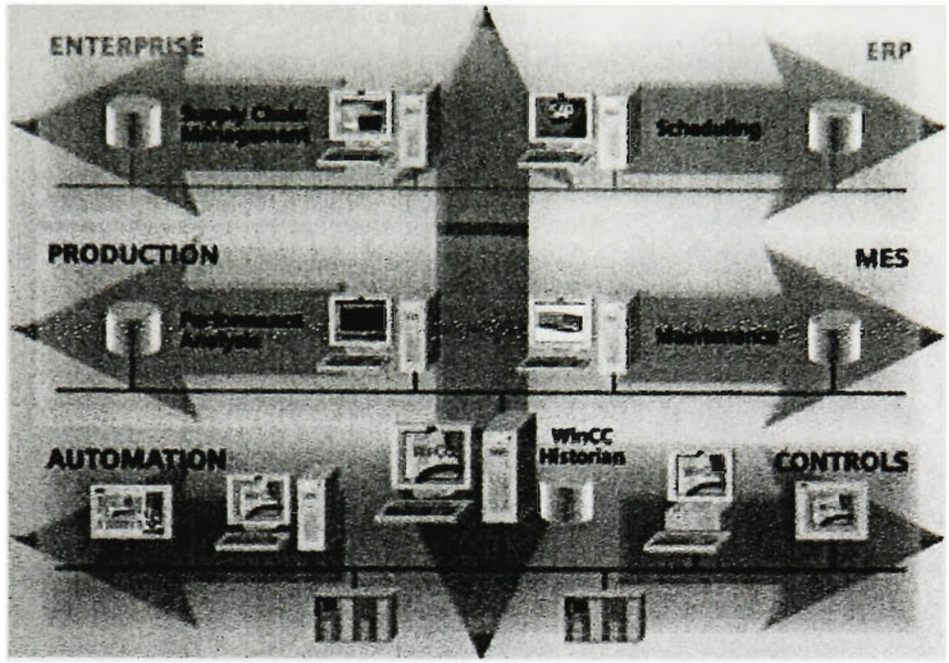


Figure 2.1: Integrating Automation systems to ERP[11]

2.2.1 Evolution of manufacturing systems[2]

The first computerized business systems were used in accounting. By the late 1960s or early 1970s, from these accounting systems evolved material requirements planning (MRP), which was intended to help manufacturers better plan material availability (Figure 2.2). By the late 1970s and early 1980s, computers were more powerful and capable of handling more data and were being used interactively by more people. MRP evolved into MRP II as shop floor reporting systems, purchasing systems and related functions were added.

MRP II didn't address the requirements of forecasting and managing demand in distribution, nor did it do much of a job in managing activities that took place there. To address the requirements in distributing and forecasting, Distribution Resource Planning (DRP) was developed. Similarly, MES and quality management evolved. Though these systems helped the manufacturers to solve business-related problems, they lacked integration with other systems. In the early 1990s, the ERP evolved and took the place of MRP II while DRP grew into Supply chain management and the shop floor solutions evolved into integrated MES systems. Meanwhile, at the control level, computers replaced manual controls. Soon, SCADA (Supervisory Control and Data Acquisition) and Graphical user interface systems became available.

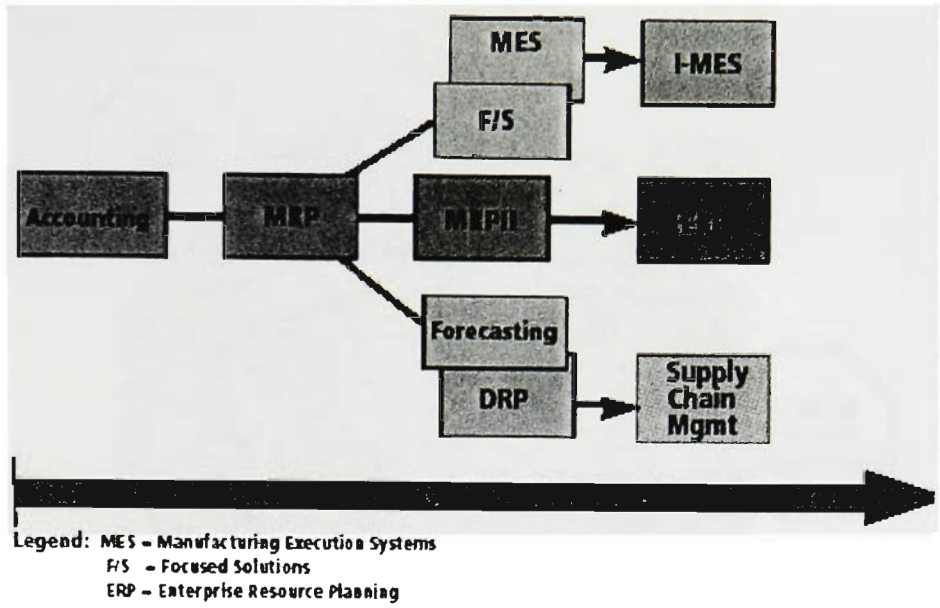


Figure 2.2 The Evolution of Manufacturing Systems

2.2.2 MES in the information systems architecture:

MES is one of several types of major information system designed for manufacturing companies. Each of these system categories include various functions and product types. Some of the other software systems that interact with MES are briefly described below

2.2.2.1 Enterprise Resource Planning (ERP):

ERP is a manufacturing information system whose focus goes beyond inventory control and distributed management activities. It consist of those systems that automate core corporate activities, such as finance, manufacturing, marketing, human resource and supply-chain management. It is a customize software system that handles the majority of enterprises' information system requirements. It is can be considered to be an optimum method of connection between different subsets of manufacturing systems via computer systems.

2.2.2.2 Supply Chain Management (SCM):

A *supply chain* is a network of facilities and distribution options that perform the functions of procurement of materials, transformation of these materials into intermediate and finished products, and the distribution of these products to customers. Supply chains exist in both service and manufacturing organizations, although the complexity of the chain may vary greatly from industry to industry and firm to firm.[5] It includes functions such as forecasting, distribution and logistics, transportation management, electronic commerce, and advance planning systems.[4]

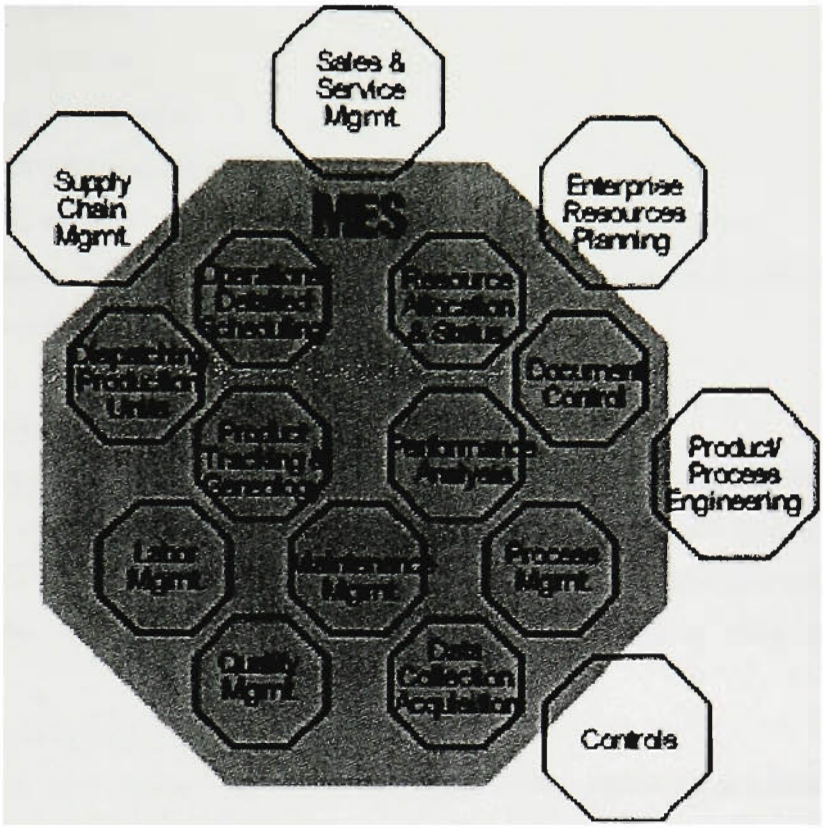


Figure 2.3 MES functional model

2.2.2.3 Supply and Service Management (SSM):

This comprises software for sales force automation, product configurations, service quoting, product returns, and so forth.[4]

2.2.2.4 Product and Process Engineering (P&PE):

Includes computer aided design and manufacturing (CAD/CAM), process modeling, and product data management (PDM).

2.2.3 MES functions:[4]

There are eleven functions of MES that link to other systems in different ways by product or needs (figure 2.3)

- Resource allocation and scheduling
- Operations/Detail scheduling
- Dispatching production units
- Document control
- Data collection/Acquisition
- Labor management

- Quality management
- Process management
- Maintenance management
- Product tracking and genealogy
- Performance analysis

Each of the functions contributes to collection of data in order to monitor, track, and estimate costs, and to control products manufactured in real time.

2.2.3.1. Resource Allocation and Status:

This manages resources including machines, tools, labor skills, materials, equipment, and other entities such as documents that must be available in order for work to start. It provides a detailed history of resources and insures that equipment is properly set up for processing and provides status in real time. The management of these resources includes reservation and dispatching to meet operation-scheduling objectives.

2.2.3.2. Operations/Detail Scheduling:

Provides sequencing based on priorities, attributes, characteristics, and/or recipes associated with specific production units at an operation. It influences factors such as shape, color sequencing, or other characteristics that, when scheduled in sequence properly, minimize set-up. It is finite and it recognizes alternative and overlapping/parallel operations in order to calculate, in detail, the exact time of equipment loading adjusted to shift patterns.

2.2.3.3. Dispatching Production Units:

Manages flow of production units in the form of jobs, orders, batches, lots, and work orders. Dispatch information is presented in the sequence in which the work needs to be done and changes in real time as events occur on the factory floor. It has the ability to alter the prescribed schedule on the factory floor. Rework and salvage processes are available, as well as the ability to control the amount of work in process at any point with buffer management.

2.2.3.4. Document Control:

Controls records/forms that must be maintained with the production unit, including work instructions, recipes, drawings, standard operation procedures, part programs, batch records, engineering change notices, shift-to-shift communication, as well as the ability to edit "as planned" and "as built" information. It sends instructions down to the operations, including providing data to operators or recipes to device controls. It might also include the control and integrity of environmental, health and safety regulations, and ISO information such as Corrective Action procedures.

2.2.3.5. Data Collection/Acquisition:

Provides an interface link to obtain the inter-operational production and parametric data from the forms and records that were attached to the production unit. The data may be collected from the factory floor either manually or automatically from equipment in an up-to-the-minute time frame.

2.2.3.6. Labor Management:

Provides a status of personnel in an up-to-the-minute time frame. Includes time and attendance reporting and certification tracking, as well as the ability to track indirect activities such as material operations and analysis from a laboratory information management system (LIMS) could also be included.

2.2.3.7. Quality Management:

Provides a real-time analysis of measurements collected from manufacturing to assure proper product quality control and to identify problems requiring attention. It may recommend action to correct the problem, including correlating the symptoms, actions and results to determine the cause. It may include SPC/SQC tracking and management of off-line inspection operations, and an analysis from a laboratory information management system (LIMS) could also be included.

2.2.3.8. Process Management:

Monitors production and either automatically corrects or provides decision support to operators for correcting and improving in-process activities. Such activities may be inter-operational and focus specifically on machines or equipment being monitored and controlled, and/or intra-operational, which is tracking the process from one operation to the next. It may include alarm management to make sure factory personnel are aware of process changes that are outside acceptable tolerances. It provides interfaces between intelligent equipment and MES, possibly through Data Collection/Acquisition.

2.2.3.9. Maintenance Management:

Tracks and directs the activities to maintain the equipment and tools, and to insure their availability for manufacturing, including scheduling for periodic or preventive maintenance. This also provides the response (alarms) to immediate problems. It maintains a history of past events to aid in diagnosing new problems.

2.2.3.10. Product Tracking and Genealogy:

Provides visibility for determining where work is at all times and its disposition. Status information may include who is working on it, as well as components, materials by supplier, lot, serial number, current production conditions, and any alarms, rework, or other exceptions related to the product. The on-line tracking function creates a historical record, as well. This streamlines the process of tracing components and usage of each end product.

2.2.3.11. Performance Analysis:

Provides up-to-the-minute reporting of actual manufacturing operations results along with comparisons to past history and expected business results. Performance results include such measurements as resource utilization, resource availability, product unit cycle time, and conformance to schedule and performance to standards. It may include SPC/SQC. Performance Analysis draws on information gathered from different functions that measure operating parameters. These results may be prepared as a periodic report or presented on-line as current evaluation of performance.[3]

2.2.4 MES function in a factory

Figure 2.4 describes the data flow between ERP (Business), MES (Execution), and Controls layers.

ERP systems work with time spans of days, weeks, months, and years. The ERP system notes product usage, customer orders, and material requirements, and sends requests to the execution (MES) layer.

MES makes wise manufacturing decisions for a company. MES develops the work instruction for the control layer. The MES systems are responsible for carrying out product manufacturing and all operations associated with the creation of those products. Product design details can be stored at the MES layer, which supplies instructions to the control layer on how to build the product. The MES system works in short time spans of one day, one shift, one hour, minute or second.[1]

Once the control parameters--such as instructions, programs, documents, software, and other manufacturing requirements for support systems--are transmitted, the control layer is then responsible for carrying out the process. The control layer works in real-time (with a time factor of 1x).

The control function uses all of the resources on the factory floor (hardware, software, and people) in a manner consistent with the goal of producing a product that meets or exceeds desired specifications.

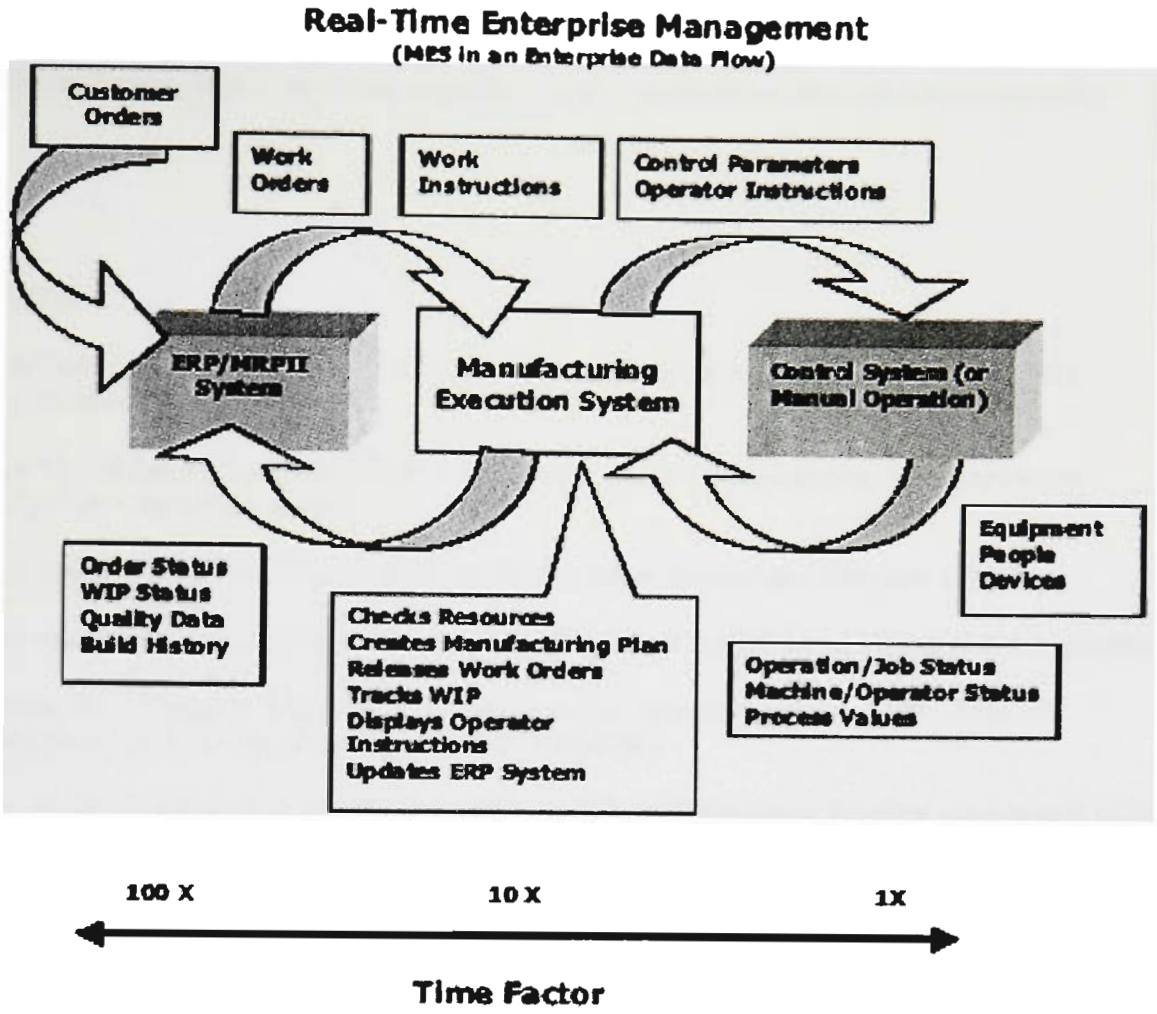


Figure 2.4 Real-Time Enterprise Management

2.3 Control Layer

The components of the Control Layer include programmable controllers, computers, robots, DCSs (Distributed Control Systems), sensing devices, Computerized Numeric Controllers, user/operator interfaces, human-machine interfaces, and intelligent input devices.

The control layer focuses on production line and process decisions. MES transmits instructions, programs, documents, software, and other manufacturing requirements for support systems to the control layer, which, in turn, uses all the hardware, software and people to carry out the process.

The shop floor (Control layer) is a constantly changing environment. A time factor of 1x (less than a second) means there are operations always occurring to refine or correct the process to maintain desired outputs. The "Equipment"," People", and "Devices" carry out the finite instructions for the process output.[3]

Thus, MES provides the real-time visibility of all plant data so that scheduling and planning software can optimize production. It provides a meaningful relationship between business data and factory data.

References:

[1]MESA International-White Paper Number 3, Controls Definition & MES to Controls Data Flow Possibilities. (February 2000).

[2]MESA International-White Paper Number 5, Execution-Driven Manufacturing Management for Competitive Advantage. (1997).

[3]MESA International-White Paper Number 4, MES Software Evaluation / Selection. (1996).

[4]MESA International-White Paper Number 6, MES Explained: A High Level Vision. (1997, September).

[5]Ganeshan, R. & Harrison, T.P. An introduction to supply chain management. (1995, May 22). <http://lcm.csa.iisc.ernet.in/scm/supply_chain_intro.html>

[6]Welti , N., & de Schepper, A. (1999). Successful sap R/3 implementation: Practical management of ERP projects. Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

[7]Kim, D. K (1996). G7 advanced manufacturing system project: System integration for FMS.Project Report, Tongil Heavy Industries Co. Ltd., Korea.

[8]Koch, C. The ABC of ERP. (2002, February 7). <www.cio.com/research/erp/edit/erpbasics.html>

[9]Siemens Automation Drives <www.siemens-industry.co.uk/automation-solutions/industrial.asp>

[10]Sober IT. (2001, March). <www.soberit.hut.fi/T-86/T-86.141/s2001/seminarslides/Automation_ERP.pdf>

[11]Siemens Business Integration. (2003) <www.ad.siemens.de/hmi/html_76/products/software/wincc/integration01.htm>

Section 3. Structure Query Language and Normalization

3.1 SQL (Structured Query Language)

SQL is a simple but powerful database language that is the standard language for many relational database systems. Some common relational database management systems that use SQL are DB2, Oracle, Sybase, Microsoft SQL, Informix etc. Although most database systems use SQL, most of them also have their own additional proprietary extensions that are most often used only on their system. However, the standard SQL commands such as "Select", "Insert", "Update", "Delete", "Create", and "Drop" can be used to accomplish almost everything that one needs to do with a database.[1]

3.1.1 Types of SQL Statements[2]

Standard SQL statements can be subdivided into 3 distinct groups

3.1.1.1 Data Definition Language (DDL) Statements:

Data Definition Language is a set of SQL commands used to create modify and delete database structures (not data). These commands wouldn't normally be used by a general user, who should be accessing the database via an application. They are normally used by the DBA (at least to a limited extent), a database designer or an application developer. These statements are immediate and are not susceptible to ROLLBACK commands. Anybody using DDL must have the CREATE object privilege and a Tablespace area in which to create objects.

Create table: creates an empty table that defining its structure.

Drop table: destroys a table and all the data it contains (but only with permission).

Alter table: changes the definition of the table (column names and types, etc.).

3.1.1.2 Data Manipulation Language (DML) statements:

Data Manipulation Language is the area of SQL that permits data changes allows you to change data within the database. It consists of only three command statement groups, they are Insert, Delete and Update.

Insert: adds a row or part of a row to a table.

Delete: removes the selected row.

Update: modifies the entries for a selected row.

3.1.1.3 Data Query Language (DQL) statements:

Data Query Language is the area of SQL that allows access to data in the database and ability to impose ordering upon it. It consists of Select statements only.

Select: The SELECT statement is the heart of SQL. It gets the data out of the database for manipulation. When SELECT is performed against a table or tables, the result is compiled into a further but temporary table, which is displayed (or received by the program).

3.2 Normalization[4]

Normalization is the process of organizing data in a database. This includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.

Redundant data wastes disk space and creates maintenance problems. If data that exists in more than one place must be changed, the data must be changed in exactly the same way in all locations. A customer address change is much easier to implement if that data is stored only in the Customers table and nowhere else in the database.

There are a few rules for database normalization. Each rule is called a "normal form." If the first rule is observed, the database is said to be in "first normal form." If the first three rules are observed, the database is considered to be in "third normal form." Although other levels of normalization are possible, third normal form is considered the highest level necessary for most applications.

In a discussion of normalization, the following concepts apply:

- **Entities** – Entities are real world objects or concepts (*customer, department, product*)
- **Relationships** – Relationship describes how two or more entities interact (*customer has property, department produces product, product has price*)
- **Tuples-** Each row, which is called a Tuple. A Tuple is an instance of an entity or relationship or whatever is represented by the relation.[5]
- **Attributes** – Attributes describe properties of entities and relationships (customer *name*, department *number, quantity* of product sold). Each column is called an Attribute.[5]
- **Primary Key**- Each row is uniquely identified by means of a primary key. Its value should be unique and not null. Depending upon the relationship it can be singular or composite.

- **Foreign Key**- An attribute in the current table that is the primary key in other table.
- **Composite key**- When a combination of attributes is used as a unique identifier, it is known as a composite key.
- **Entity Integrity-** The entity integrity rule states that for every instance of an entity, the value of the primary key must exist, must be unique, and must not be null. Without entity integrity, the primary key could not fulfill its role of uniquely identifying each instance of an entity.
- **Referential Integrity-** The referential integrity rule states that every foreign key value must match a primary key value in an associated table. Referential integrity ensures that users can correctly navigate between related entities.

3.2.1 The Raw Database[3]

A database that is not normalized may include data that is contained in one or more different tables for no apparent reason. This could be bad for security reasons, conservation of disk space usage, speed of queries, efficiency of database updates, and, perhaps most importantly, data integrity. A database before normalization is one that has not been broken down logically into smaller, more manageable tables. Figure 3.1 illustrates the database used for this book before it was normalized.

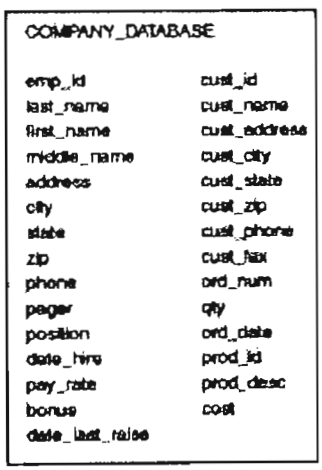


Figure 3.1 The Raw Database

3.2.2 The Normal Forms[3]

Normal form is a way of measuring the levels, or depth, to which a database has been normalized.

The following are the three most common normal forms in the normalization process:

- The first normal form
- The second normal form
- The third normal form

Of the three normal forms, each subsequent normal form depends on normalization steps taken in the previous normal form. For example, to normalize a database using the second normal form, the database must first be in the first normal form.

3.2.2.1 The First Normal Form[3]

The objective of the first normal form is to divide the base data into logical units called tables. When each table has been designed, a primary key is assigned to most or all tables. Figure 3.2, illustrates how the raw database, shown in the previous figure, has been redeveloped using the first normal form.

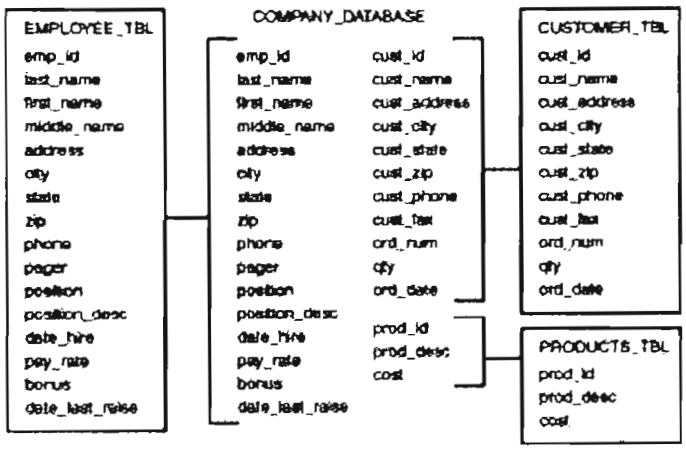


Figure 3.2 The First Normal form

You can see that to achieve the first normal form, data had to be broken into logical units, each having a primary key and ensuring that there are no repeated groups in any of the tables. Instead of one large table, there are now smaller, more manageable tables: EMPLOYEE_TBL, CUSTOMER_TBL, and PRODUCTS_TBL. The primary keys are normally the first columns listed in a table, in this case: EMP_ID, CUST_ID, and PROD_ID.

3.2.2.2 The Second Normal Form[3]

The objective of the second normal form is to take data that is only partly dependent on the primary key and enter that data into another table. Figure 3.3 illustrates the second normal form.

According to the figure, the second normal form is derived from the first normal form by further breaking down two tables down into more specific units.

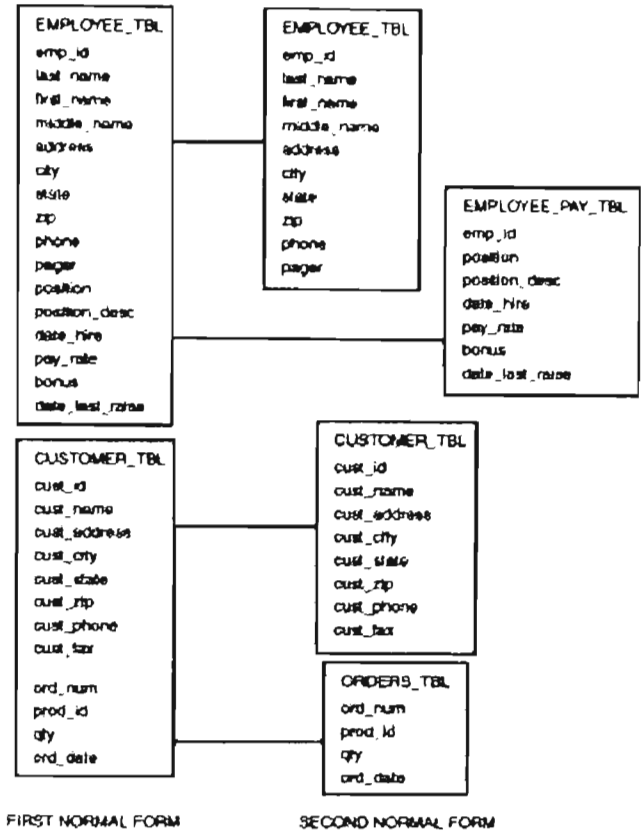


Figure 3.3 The second normal form

EMPLOYEE_TBL is split into two tables called EMPLOYEE_TBL and EMPLOYEE_PAY_TBL. Personal employee information is dependent on the primary key (EMP_ID), so that information remained in the EMPLOYEE_TBL (EMP_ID, LAST_NAME, FIRST_NAME, MIDDLE_NAME, ADDRESS, CITY, STATE, ZIP, PHONE, and PAGER). On the other hand, the information that is only partly dependent on the EMP_ID (each individual employee) is used to populate EMPLOYEE_PAY_TBL (EMP_ID, POSITION, POSITION_DESC, DATE_HIRE, PAY_RATE, DATE_LAST_RAISE). Notice that both tables contain the column EMP_ID. This is the primary key of each table and is used to match corresponding data between the two tables.

CUSTOMER_TBL is split into two tables called CUSTOMER_TBL and ORDERS_TBL. What took place is similar to what occurred in the EMPLOYEE_TBL. Columns that were partly dependent on the primary key were directed to another table. The order information for a customer is dependent on each CUST_ID, but does not directly depend on the general customer information in the original table.

3.2.2.3 The Third Normal Form[3]

The third normal form's objective is to remove data in a table that is not dependent on the primary key. Figure 3.3 illustrates the third normal form.

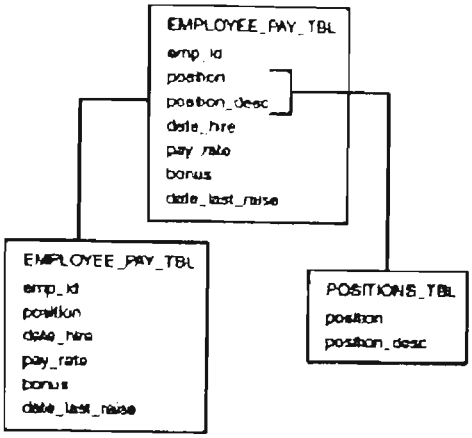


Figure 3.4 The third normal form

Another table was created to display the use of the third normal form. EMPLOYEE_PAY_TBL is split into two tables, one table containing the actual employee pay information and the other containing the position descriptions, which really do not need to reside in EMPLOYEE_PAY_TBL. The POSITION_DESC column is totally independent of the primary key, EMP_ID.

3.2.3 Benefits of Normalization[3]

Normalization provides numerous benefits to a database. Some of the major benefits include the following:

- Greater overall database organization
- Reduction of redundant data
- Data consistency within the database
- A much more flexible database design
- Improved database security

Organization is brought about by the normalization process, making the job easier for everyone, from the user who accesses tables to the database administrator (DBA) who is responsible for the overall management of every object in the database. Data redundancy is reduced, which simplifies data structures and conserves disk space. Because duplicate data is minimized, the possibility of inconsistent data is greatly reduced. For example, in one table an individual's name could read STEVE SMITH, whereas the name of the same individual reads STEPHEN R. SMITH in another table. A database that has been normalized and broken into smaller tables provides more flexibility as far as modifying existing structures. It is much easier to modify a small table with little data than to modify one big table that holds all the vital data in the database. Lastly, security is also provided in the sense that the DBA can grant access to limited tables to certain users. Security is easier to control when normalization has occurred.

Data integrity is the assurance of consistent and accurate data within a database.

3.2.4 Referential Integrity[3]

Referential integrity simply means that the values of one column in a table depend on the values of a column in another table. For instance, in order for a customer to have a record in the ORDERS_TBL, there must first be a record for that customer in the CUSTOMER_TBL table. Integrity constraints can also control values by restricting a range of values for a column. The integrity constraint should be included in the table's creation. Referential integrity is typically controlled through the use of primary and foreign keys.

In a table, a foreign key, normally a single field, directly references a primary key in another table to enforce referential integrity. In the preceding paragraph, the CUST_ID in the ORDERS_TBL is a foreign key that references the CUST_ID in the CUSTOMER_TBL.

3.2.5 Drawbacks of Normalization[3]

Although most successful databases are normalized to some degree, there is one substantial drawback with of a normalized database: reduced database performance. The acceptance of reduced performance requires the knowledge that when a query or transaction request is sent to the database, there are negative factors involved, such as CPU usage, memory usage, and input/output (I/O). A normalized database requires much more CPU, memory, and I/O to process transactions and database queries than does a denormalized database. A normalized database must locate the requested tables and then join the data from the tables to either get the requested information or to process the desired data.

Reference:

[1]Jupitermedia Corporation, (2003)
<www.sqlcourse.com/intro.html>

[2]Angell Developments
<www.ilook.fsnet.co.uk/ora_sql/sqlmain.htm>

[3]Plew, R.R. & Stephens, R.K.. (2000). *Sams Teach Yourself SQL in 24 Hours. Sams*

[4]Microsoft Knowledge Base Article – 209534(2002)
<support.microsoft.com/default.aspx?scid=KB;en-us;q209534>

[5]Brown, C.E.Database Learning Module. (2001).
<www2.bus.orst.edu/faculty/brownc/lectures/db_tutor/index.htm>

Section 4. Client-Server methodology for Data Management

4.1 Introduction[1]

Today, stand-alone computers have been replaced by computers in networks for most processing tasks. The use of multiple computers linked by a communications network for processing is called distributed processing. In contrast with centralized processing, in which all processing is accomplished by one large central computer, distributed processing works among PCs, minicomputers, and mainframes that are linked together.

One widely used form of distributed computing is client/server computing. Client/server systems operate on network environments, splitting the processing of an application between a front-end client and a back-end processor. Client/server describes the relationship between two computer programs where one program is where the client makes the request from another program, the server, which fulfills the request. Figure 4.1. Illustrates the client-server computing concept.

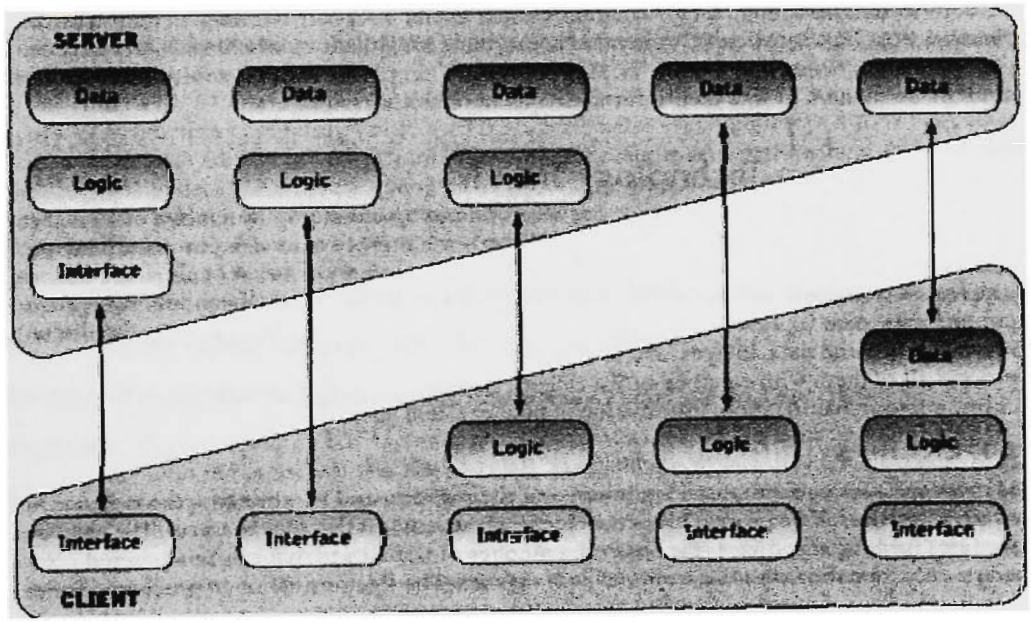


Figure 4.1 Client/Server-computing[1]

The several client/server architectures that have evolved can be distinguished by the distribution of application logic components across clients and servers. As illustrated in Figure2, the application logic consists of three components.

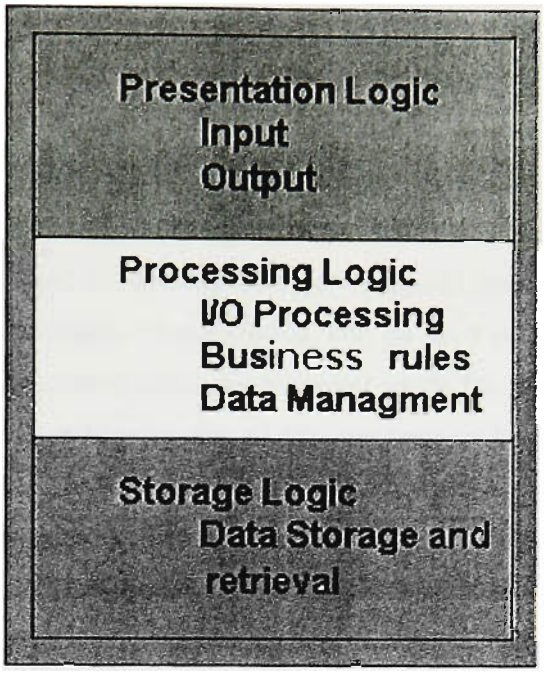


Figure 4.2 Application logic components[2]

The first is the "presentation logic component" (See Figure 4.2) .The presentation logic is concerned with managing the graphical user interface. It is responsible for formatting and presenting data on the users screen or other output devices, and for managing user input from keyboard or other input devices.[2]

The second component is the processing components also know as the Business service layer. It is responsible for implementing business rules that are specific to a company or procedure and act as a bridge between the presentation layers. It also handles data processing logic, business rules logic, and data management logic. Data processing logic includes activities such as data validation and identification of process errors. Business rules that have not been coded at the DBMS (DataBase Management System) level may be coded in the processing component.[2] Data management logic is where the particular records that are needed for a particular transaction are determined.

The third component is storage; it is responsible for actual retrieval or storage of data from or to the physical device.

4.2 File Server architectures

The original PC networks were based on a file server, where the data manipulation occurs at the desktop where the data was requested. The client handled the presentation logic, processing logic and much of the storage logic. By contrast File server is a device that manages files operations and is shared by each of the client PCs attached to a LAN (Local Area Network). Each of theses files servers acts as an additional hard disk for each of the client PCs. Programs on the PC refers to accessing files on this hard disk by typical path specification. With a file server, each client PC may be called a fat

client, one where most of the process occurs on the client PC rather than on a server.[2] Refer to figure 4.3

In these types of systems, clients run the DBMS software. When the user request some data, for example using the SQL (Structured Query Language), the DBMS at the client side decides which files it needs from the server to process it. These files are then requested from the server. Thus, there is one database but many concurrent copies of the DBMS, one on each of the active PCs. All the data manipulation is performed at the client workstation and not on the file server. Therefore, a files server acts only as a storage device. The client handles all data management functions.

In the 1990s, PC LAN (local area network) computing changed because the capacity of the file sharing was strained as the number of online users grew (one LAN can only satisfy about 12 users simultaneously) and graphical user interfaces (GUIs) became popular (making mainframe and terminal displays appear out-of-date). PCs are now being used in client/server architectures.[3]

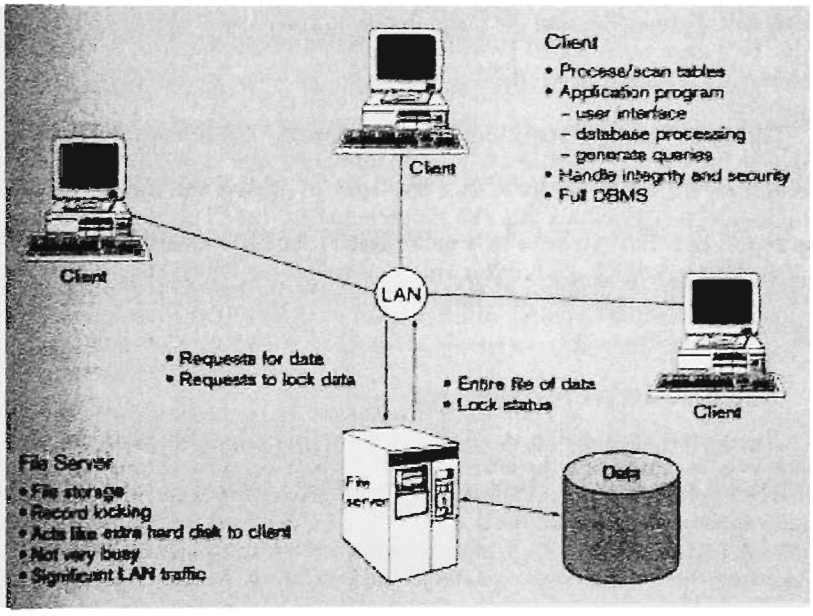


Figure 4.3 File server model[2]

These are the limitations when using file servers on LAN[2]:

1) There is a considerable movement across the network. A full functional copy of DBMS is required at the client side; even if few records are required the whole file containing them must be passed. Thus, the server does very little work and the client is busy performing data manipulation. These types of systems place considerable burden on the client workstation and create a high network traffic load.
2) Since the client workstation does the processing, each end user whose work requires heavy and frequent database access needs a high-performance computer with plenty of memory. This means there is less room in the memory for application programs on the client PC. Nonetheless, increasing RAM on the PC will improve performance by increasing the total

amount of data that can reside on the PC while a transaction is being processed. The file server does not need much RAM and need not be a very powerful PC since it does little work.

3) The DBMS copy in each workstation must manage the shared database integrity. In addition, each application program must recognize, for example, locks and take care to initiate the proper locks. The developers must understand how their application will interact with the DBMS's concurrency, recovery and security controls, and sometimes must program such controls into their applications.

4.3 Database Server Architecture

Database Server Architecture (also known as "Two tier client/server architecture") was developed in the 1980s from the file server software architecture design. In this type of architecture, the client workstation is responsible for managing the user interface, including presentation logic, data processing logic and business rules logic, and the database server is responsible for database storage, access and processing.[2] With this type of system, the data management logic is split between the client and server. The LAN traffic is reduced because the DBMS is placed in the server, and only those records are transmitted that match the request criteria. In response to a user action, the client sends an SQL request to the server that executes it and returns or modifies data in the database appropriately. The Central DBMS function is referred to as a back-end function, whereas the application functions running on the client side are referred as front-end programs. Figure 4.4 shows typical database server architecture.

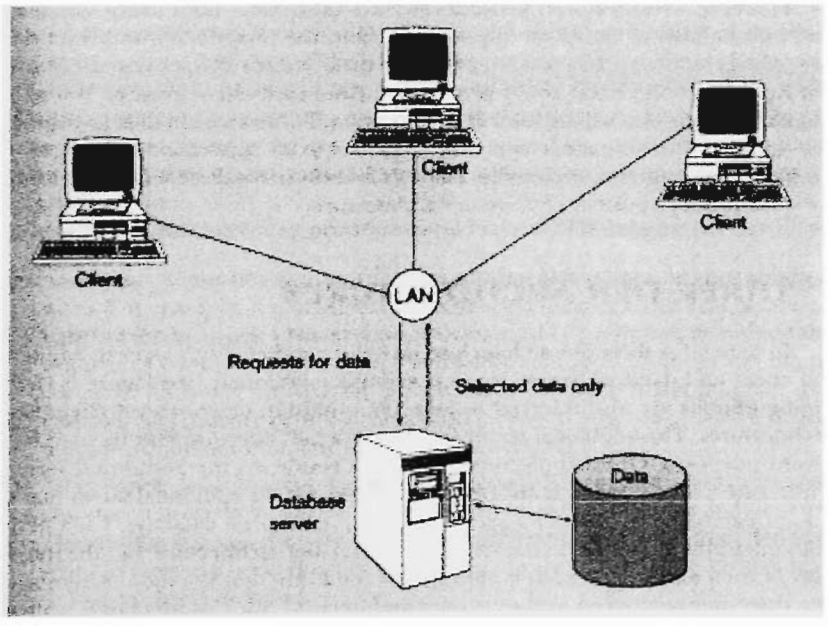


Figure 4.4 Database Server Architecture[2]

Database Server Architecture has several advantages:

1) With this type of architecture only the database server requires processing power adequate to handle the database since the database is stored on the server and not on the client.
2) The database server can be tuned to optimize database-processing performance.[2]
3) The network traffic is considerably reduced because less data is transmitted across the LAN.
4) User authorization, integrity checking, data dictionary maintenance and query and update processing are all performed in one location, on the database server. [2]

Limitations of Database Server Architecture are:

1) The Client side is rather "fat" due to the business logic that resides there.
2) Changes to the business logic at the server also require changes to presentation layer. This will require that each client be upgraded separately.
3) Two-tier architecture provides limited flexibility in moving program functionality from one server to another without manually regenerating procedural code.

These drawbacks to database server architecture have led to the popularity of three-tier architecture.

4.4 Three-Tier Architecture

Three-tier software architecture emerged in the 1990s to overcome the limitations of the two-tier architecture (See Figure 4.5). A middle tier is added between the user's system interface client environment and database management server layer. Often application programs reside on an additional server, which is referred to as an application server. But the additional server may hold a local database while another server holds the enterprise database.[2]

There are a variety of ways of implementing this middle tier, such as transaction processing monitors, mail servers, or application servers. The middle tier can perform queuing, application execution. For example, if the middle tier provides queuing, the client can deliver its request to the middle layer and disengage because the middle tier will access the data and return the answer to the client. In addition the middle layer adds scheduling and prioritization for work in progress.[3] In many situations most business processing occurs on the application server rather than on the client workstation or database server, resulting in a thin client.

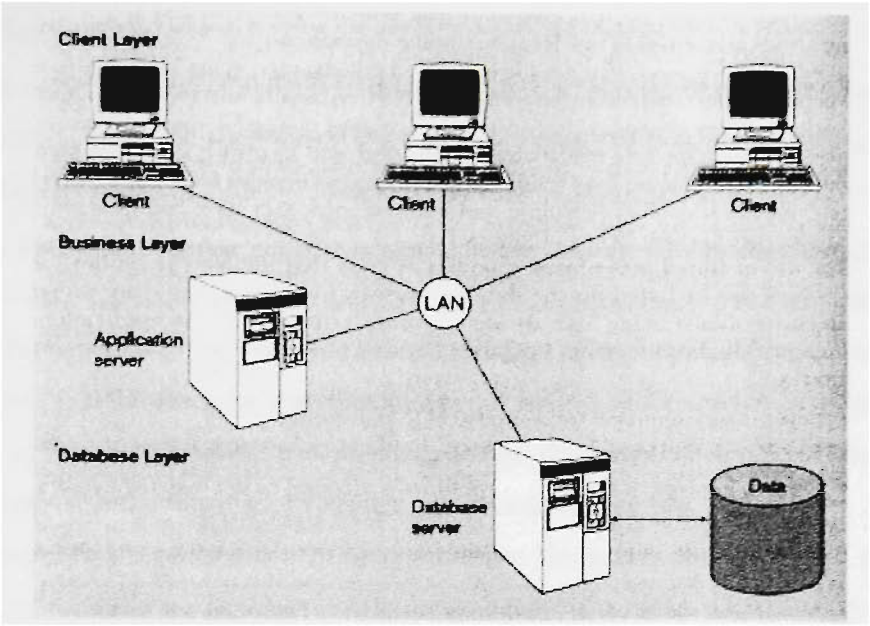


Figure 4.5 Three-tier Architecture[2]

Three-Tier Architecture is also referred to as n-tier, multi-tier, or enhanced client/server architectures. Three-Tier Architecture can provide several benefits[2]:

1) **Scalability:** Three-Tier Architectures are more scalable than two-tier architectures. For example, the middle tier can be used to reduce the load on a database server by using a transaction-processing (TP) monitor to reduce the number of connections to a server, and additional application servers can be added to distribute application processing. A TP monitor is a program that controls data transfer between clients and servers in order to provide a consistent environment for online transactions processing (OLTP).
2) **Technological flexibility:** It is easy to change DBMS engines, though triggers and stored procedures will need to be rewritten with a three-tier architecture. The middle tier can even be moved to a different platform.
3) **Lower long-term cost:** Use of off-the-self components or services in the middle tier can reduce costs, as can substitution of modules within an application rather than an entire application.
4) **Better match of systems to business needs:** New modules can be build to support specific business needs rather building more general, complete applications.
5) **Improve customer service:** Multiple interfaces on different clients can access the same business process.
6) **Competitive advantage**: The ability to react to business changes quickly by changing small modules of code rather than entire applications can be used to gain a competitive advantage.
7) **Reduce risk**: Again, the ability to implement small modules of code quickly and to combine them with code purchased from vendors limits the risk assumed with a large-scale development project.

4.5 Client/Server issues

In order to succeed, client/server projects should address a specific business problem with a well-defined technology and cost parameters. Certain areas should be carefully addressed in order to improve the chances for building a successful client/server application: [2]

- **Accurate business problem analysis:** Just as in the case of other computing architectures, it is critical to develop a sound application design and architecture for new client/server systems. Developers, tendency to pick the technology and then fit the application to it seems to be more pronounced in the strong push toward client/server environments that has occurred in the last six years. It is more appropriate to accurately define the scope of the problem, determine the requirements, and then use that information to select the technology.
- **Detail architecture analysis**: It is also important to specify the details of the client/server architecture. Building a client/server solution involves connecting various company components that may not work together easily. One of the most touted advantages of client/server computing, the ability to accept an open systems approach, can be detrimental if the heterogeneous components chosen are difficult to connect. Besides specifying the client workstations, servers, network, and DBMS, analysts also should specify network infraructure, the middleware layer, and the application development tools to be used. At each juncture, analysts should take steps to assure that the tools will connect with the middleware, database, network, and so forth.
- **Avoiding tool-driven architecture:** As above, the designer should determine the project requirements before choosing software tools, and not the reverse. Choosing a tool first and then applying it to the problem risks having a poor fit between the problem and the tool. Tools selected in this way are likely to have been chosen based on an emotional appeal rather than on the appropriate functionality of the tool.
- **Achieving appropriate scalability:** A multi-tier solution allows client/server system to scale to any number of users and handle diverse processing loads. But multi-tiered solutions are significantly more expensive and difficult to build. Architects should avoid moving to a multi-tier solution when its not really needed. Usually multi-tier architecture makes sense in environments of more than 100 concurrent users. Smaller, less intense environments can frequently run more efficiently on traditional two-tier systems, especially if triggers and procedures are used to manage the processing.
- **Appropriate placements of services:** Again, a careful analysis of the business problem being addressed is important when making decisions about the placements of processing services. The move towards thin clients and fat servers is not always the appropriate solution. Moving the application logic to server, thus creating a fat server, can affect capacity, as end users all attempt to use the application now located on the server. Fat servers do tend to reduce network load because the processing takes place close to the data, and they do lessen the need for powerful clients. Understanding the business problem intimately should help the architecture to distribute the logic appropriately.

- **Network analysis:** The most common bottleneck in the distributed systems is still then network. If the network is insufficient to handle the amount of information that must pass between client and server, response time will suffer badly, and the system is likely to fail.

Reference:

[1] Laudon K C. (February 2003) *Management information systems*, New Jersey: Prentice Hall,

[2] Mcadden F R.(1998 August) *Modern database management*, fifth edition, Addison-Wesley

[3] Carnegie Mellon University (2003 September)
<www.sei.cmu.edu/str/descriptions/clientserver_body.html>

Section 5. Open Database Connectivity

5.1 Introduction to ODBC

Corporations typically have applications and data residing on diverse platforms and database management systems due to historical, strategic or technological reasons.[1]

The Microsoft Open Database Connectivity (ODBC) interface is a C programming language interface that makes it possible for applications to access data from a variety of database management systems (DBMS).[2]

ODBC is based on a specification, which was developed by a consortium of over 40 companies (members of SQL Access Group and others), and has broad support from application and database vendors. It provides an open, vendor-neutral way of accessing data stored in a variety of proprietary personal computers, minicomputers and mainframe databases.[1]

ODBC provides many significant benefits to developers and end users:[2]

- ODBC allows corporations to continue to use their existing database management systems, while enabling access to their data by common applications.
- ODBC benefits users as more end-user applications connect to additional data sources, making the vast volumes of corporate data more readily available.
- ODBC allows users to access data in more than one data storage location (for example, more than one server) from within a single application.
- ODBC allows users to access data in more than one type of DBMS (such as DB2, Oracle, and dBASE) from within a single application.
- ODBC provides a standard, open, and vendor-neutral API.
- ODBC is based upon the SQL Access Group (SAG) Call Level Interface (CLI) and provides a standard SQL language based upon ANSI standards.
- ODBC allows corporations and software vendors to protect their investments in existing DBMSs.

5.2 ODBC Architecture

ODBC defines a database independent standard, which is how applications can interact with a database. A Driver Manager sits between the application and the database's specific drivers. In Windows, the Driver Manager and the drivers are implemented as dynamic-link libraries (DLLs).[3]

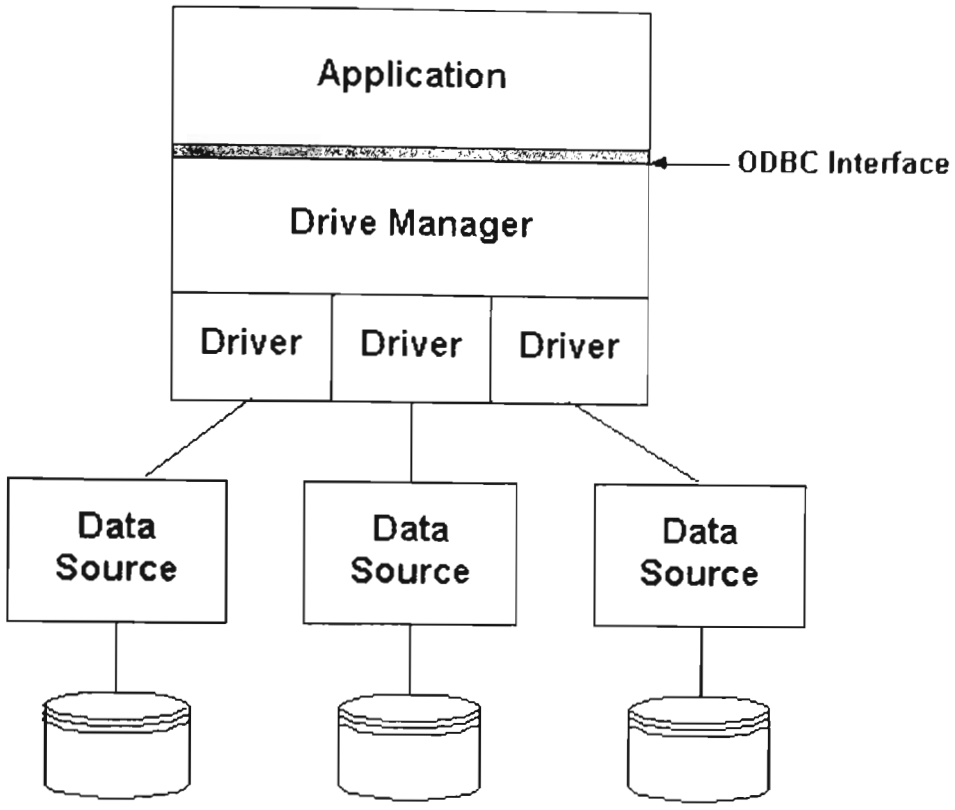


Figure 5.1 ODBC Architecture

The architecture has four components:

5.2.1 Application:

An application (e.g., MS Office or MS Access) using the ODBC interface performs the following tasks:

- Requests a connection, or session, with a data source.
- Sends SQL requests to the data source.
- Defines storage areas and data formats for the results of SQL requests.
- Requests results.
- Processes errors.
- Reports results back to a user, if necessary.
- Requests commit or rollback operations for transaction control.
- Terminates the connection to the data source.[5]

5.2.2 Driver Manager:

Driver Manager is a *Dynamic Link Libra y* (DLL) that loads and unloads drivers on behalf of an application. The Driver Manager also performs the following:

- Uses the ODBC.INI file or registry to map a data source name to a specific driver dynamic-link library (DLL).
- Processes several ODBC initialization calls.

- Provides entry points to ODBC functions for each driver.
- Provides parameter validation and sequence validation for ODBC calls.[5]

5.2.3 Driver:[1]

The driver, developed separately from the application, sits between the application and the network. Specific tasks performed by drivers include the following:

- Connecting to and disconnecting from the data source.
- Checking for function errors not checked by the Driver Manager.
- Initiating transactions (this is transparent to the application).
- Submitting SQL statements to the data source for execution. The driver must modify ODBC SQL to DBMS-specific SQL (often limited to replacing escape clauses defined by ODBC with DBMS-specific SQL).
- Sending data to and retrieving data from the data source, including converting data types as specified by the application.

5.2.4 Data Source:[2]

This is the data the user wants to access and its associated operating system, DBMS, and network platform (if any) used to access the DBMS. Applications are not limited to communicating through one driver. A single application can make multiple connections, each through a different driver, or multiple connections to a similar source through a single driver.

5.3 Data Source Name (DSN)

The Data Source Name stores information about how to connect to a particular ODBC database. The purpose of a Data Source is to gather all the technical information needed to access the data--the driver name, network address, network software, and so on--into a single place and make the data access transparent to the user.

For example, a user should be able to look at a list of database offerings that could include employees, payroll, and inventory, and then choose "employee" from the list. The application would then connect the employee data, all without the user knowing where the employee data resides or how the application got to it.[3] The ODBC driver manager uses the information in the DSN to establish and manage the connection. There are two types of DSNs. Both types store information needed to make a connection to a database server, but they store it in different locations. Their descriptions follow.

5.3.1 File DSN:

A File Data Source can be shared among all users who have the same drivers installed. These data sources need not be user-dedicated or local to a computer. File Data Sources, which are stored in a file, and allow

connection information to be used repeatedly by a single user or shared among several users. When a File Data Source is used, the Driver Manager makes the connection to the Data Source using the information from a .dsn file. This file can be manipulated like any other text file. A File Data Source does not have a Data Source name, as does a machine data source, nor is it registered to any user or system.

5.3.2 Machine Data Source:

Machine data sources are stored on the system with a user-defined names. Associated with the Data Source name is all the information the database driver needs to connect to the Data Source and that the Driver Manager needs to coordinate all the Data Sources and drivers. There are two machines-data source types, "user" and "system".[3]

5.4 Visual FoxPro Connections

In order to access any external data through ODBC, one must make a connection. Visual FoxPro allows connections to be stored permanently in a database, or created on the fly using the SQL pass-through commands. Figure 5.2 shows a VFP connection design that requires the connection parameters to be entered. If the DSN is already created, it is listed in the data source list box. If the SQL connection string option is chosen, then the connection string requires user input.

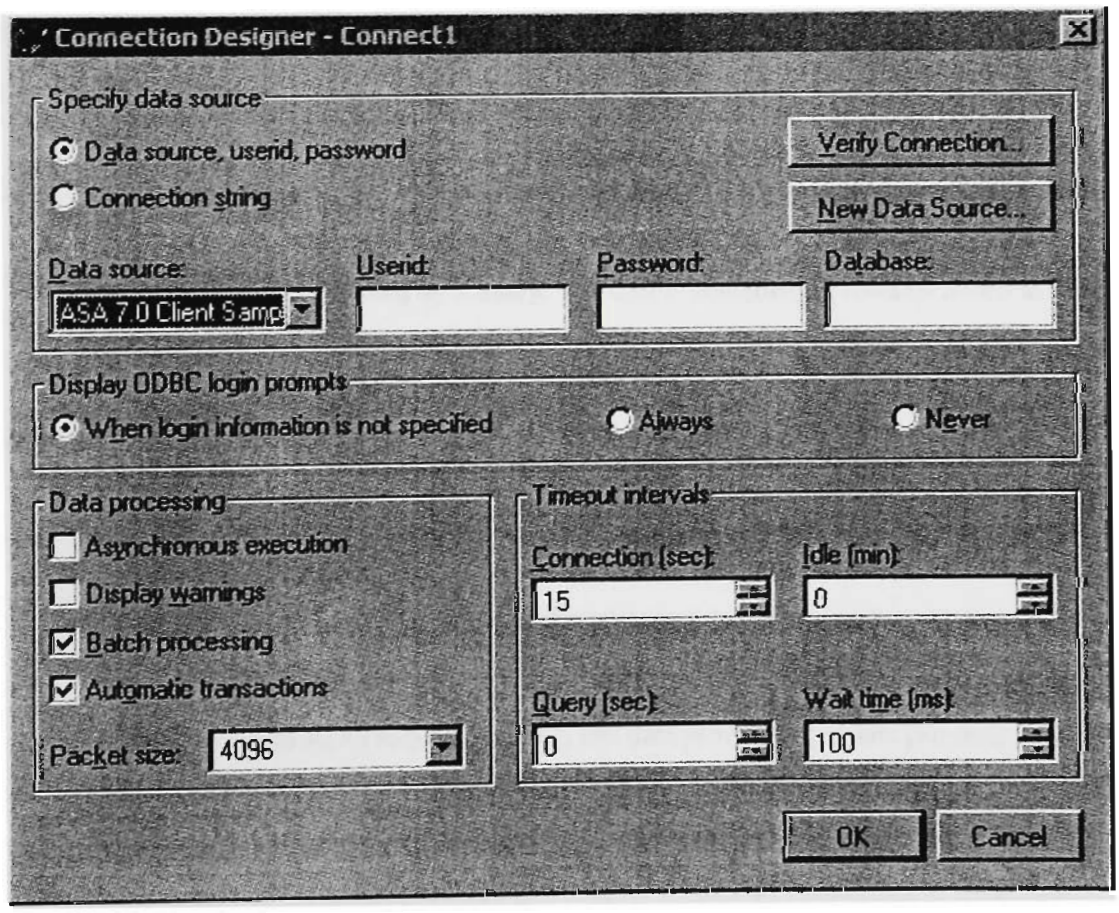


Figure 5.2 VFP Connection Designer

5.5 Remote Views in Visual FoxPro

Visual FoxPro has a built-in powerful database engine. It also has a native access to DBF files, either to free tables or to tables registered with a database container. For connecting to data in other formats, users access the ODBC data source, ADO or RDO to get data that is stored outside VFP. Using VFP to connect live to external data using ODBC, it is called ***Remote Data***.[4]

Microsoft has built in two fundamental ways to connect to remote data. The easier, friendlier way is with the ***Remote View***. VFP views are temporary tables created by permanently-stored SQL statements as an object in the database container, just like a connection.[4] This is a view that has a remote data source as its base table, rather than a local DBF. In addition, though, a set of commands that expose much of the ODBC API, called SQL pass-though commands, can be used to send SQL strings to the host data source.

5.6 ActiveX Data Objects (ADO)[6]

ActiveX Data Objects (ADO) are part of the Universal Data Access (UDA) technology that provides access to information across the enterprise. Like its predecessor, Open Database Connectivity (ODBC), UDA provides a common interface for communication with SQL databases.

Two objects are particularly important to administrators who need to connect to databases when writing scripts. These two objects, the Connection object and the Recordset object, are explained in more detail in Table 5.1.

Object	Description
Connection	Manages the connection between the script and the database. The Connection object is used to open the database but does not return any data. The Recordset object is used to return data.
Recordset	Contains the data returned by a query. The data is contained in rows (referred to as records) and columns (referred to as fields). Each column is stored in a Field object in the Recordset Fields collection. Recordsets are returned by using SQL commands such as "SELECT * FROM Hardware". A recordset can represent all the records in a database or a subset, depending on how the SQL queries are constructed.

Table 5.1: ADO Objects

Connection to a database via an ADO requires an OLE DB data provider; this data provider serves as the mechanism to connect to a particular type of database (SQL Server, Microsoft Access, Active Directory, and so on). Although it is possible to connect directly to a database by including the provider name and the path to the database as part of the connection string, a simpler method of connecting to a database is to create a data source name (DSN) for the database.

5.6.1 Connecting to a Database

To connect to a database by using ADO, the same standard procedure is followed, regardless of the type of target database. The script must perform as follows:

1. Create instances of the Connection and Recordset objects.
2. Use the Connection object Open method to connect to the DSN for the database.
3. Use the Recordset object Open method to retrieve data from the desired table within the database.

The Recordset object Open method requires both a standard SQL command (for example, "SELECT * FROM Inventory") and parameters specifying CursorLocation, CursorType, and LockType. These parameters are described in more detail in Table 5.2.

Property	Description
CursorLocation	Data structure that holds the results of any query made. Cursors can be stored either on the server or on the client. In general, it is better to store the cursor on the client; this tends to improve performance (because the data is stored locally) and places less strain on the database server. To set the CursorLocation to the client, the value of the constant adUseClient is set to 3.
CursorType	Allows browsing a recordset in different ways: some cursors allow users to move backward and forward in a recordset, while other cursors limit users to moving forward only. If the CursorLocation is set to the client, the CursorType must be set to adOpenStatic (value = 3). This type supports scrolling forward and backward in the recordset but does not show changes made by other users. This is because the cursor is operating on data cached on the client computer rather than directly from the database.
LockType	Determines how (and if) a recordset can be updated. Recordsets can be set to read-only, or they can be configured to allow updates. For most scripts, the LockType can be set to adLockOptimistic (value = 3). With this setting, the record being edited is not locked (that is, no restrictions are placed on another user accessing that record) until the Update method is called.

Table 5.2 Parameters for the Recordset Object Open Method

When you are finished with the database connection and the recordset, use the Close method to close both objects.

5.6.2 Scripting Steps

The script must perform the following steps:

1. Create three constants (adOpenStatic, adLockOptimistic, and adUseClient) and set the value of each to 3. These constants will be used to configure the CursorLocation, CursorType, and LockType for the connection.
2. Create an instance of the ADO Connection object (ADODB.Connection). The Connection object makes it possible to issue queries and other database commands.
3. Create an instance of the ADO Recordset object (ADODB.Recordset). The Recordset object stores the data returned from a query.
4. Use the Connection object Open method to open the database with the DSN Inventory.
5. Set the CursorLocation to 3 (client side) by using the constant adUseClient.
6. Use the Recordset object Open method to retrieve all the records from the Hardware table.

   The Open method requires four parameters:

   - The SQL query ("SELECT * FROM Hardware")
   - The name of the ADO connection being used (objConnection)
   - The cursor type (adOpenStatic)
   - The lock type (adLockOptimistic)

7. Close the recordset. In an actual production script, of course, users would probably do more with a recordset than just opening and closing it.
8. Close the connection.

References:

[1]Marxmeier Software. (1997).
<www.msede.com/products/sqlodbc_d.htm>

[2]MSDN Library. (2003).
<www.msdn.microsoft.com/library/>

[3]Data Access Worldwide. ODBC: A technical overview. (January 2001).
<www.dataaccess.com/whitepapers/odbc.htm>

[4]Paddock, R.,Talmage, R., Petersen, J.V., Ranft, E>, & Peterson, J. (1996, October). *Visual Foxpro 5 Enterprise Development*. Premier Press.

[5]Object Tools, Inc. (2000).

<www.object-tools.com/manuals/lib/dale/ch_03.html>

[6] Microsoft® Windows® 2000 Scripting Guide. (2000).

<www.microsoft.com/technet/treeview/default.asp?url=/technet/scriptcenter/scrguide/sas_ent_eici.asp>

Section 6. Siemens WinCC Features

This section discusses the important capabilities of WinCC that will be helpful for capturing real-time data from the CAMCELL application into MES database. The companion thesis has already described the architecture of the Simatic Automation system. This thesis section, elaborates on User Archiving and Global Script editors of WinCC. It also discusses a range of Functions and Actions of WinCC. Next, it will describe the Open architecture of WinCC that makes it capable of exchanging information with the enterprise and factory levels of the MES database.

6.1 Functions & Actions[1]

WinCC makes it possible to use functions and actions to create processes in the WinCC project dynamic. These functions and actions are written in the ANSI-C language.

6.1.1 Functions & Actions - Differences

Actions are started by a trigger, i.e., an initiating event. Functions do not have a trigger and are used as components of actions as well as in Dynamic Dialogs, in Tag Logging and in Alarm Logging.

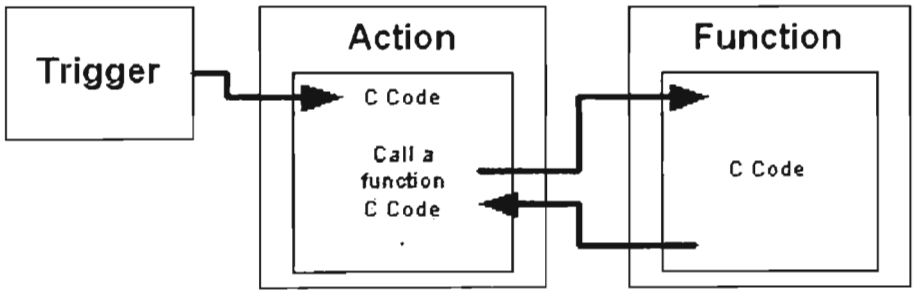


Figure 6.1: Actions & Functions

6.1.2 Types of Triggers

The following types of triggers are available:

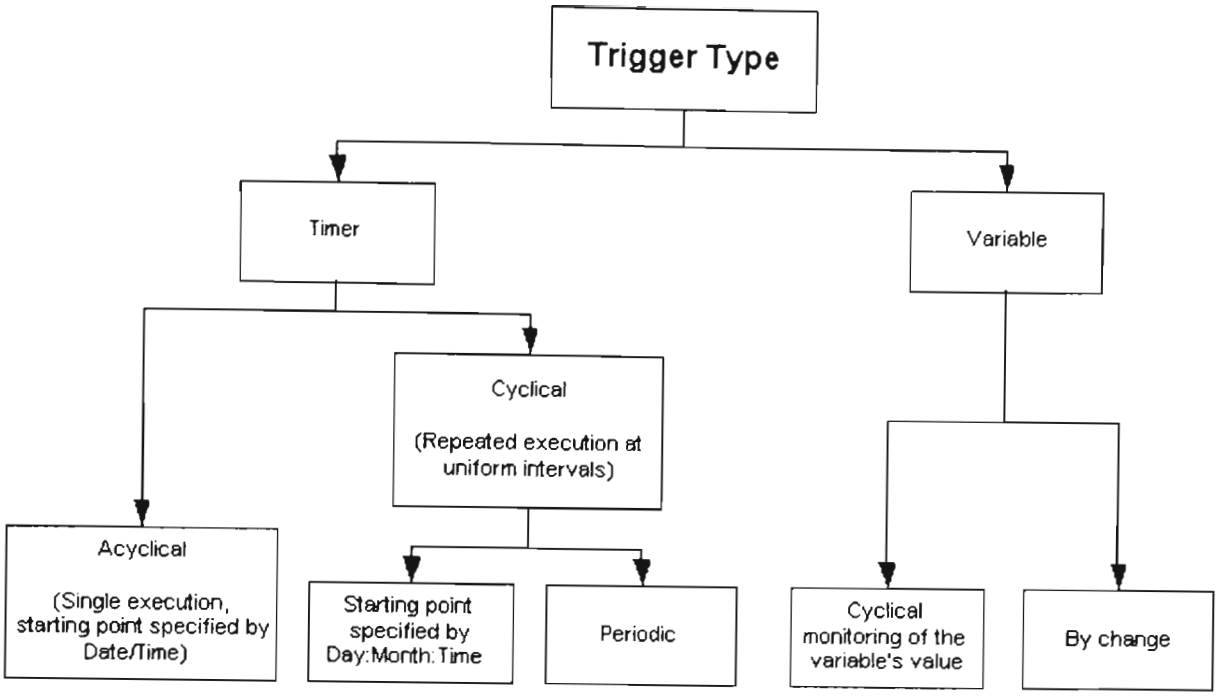


Figure 6.2: Types of Triggers

6.1.3 Outline of the functions and actions

The diagram provides an overview of the range of functions and actions:

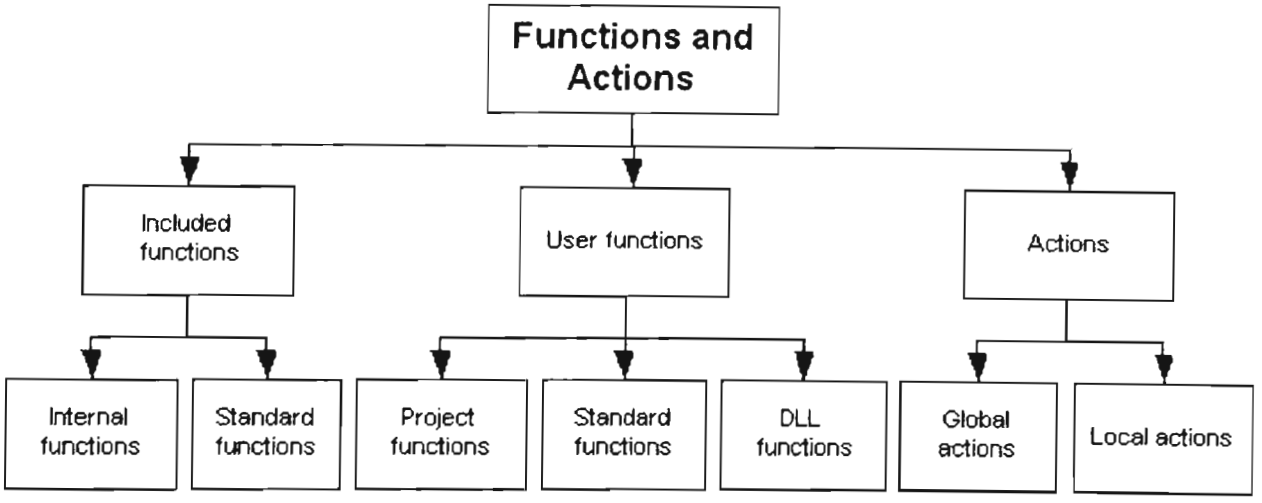


Figure 6.3: Range of Functions and Actions

6.1.4 Types of Triggers

Actions are used for picture-independent background tasks, e.g. the daily printing of a report, the monitoring of tags or the execution of calculations.

Functions are pieces of code that can be used in several locations but are defined only in one place. WinCC includes a multitude of functions. Furthermore, users can also write their own functions and actions.

The included standard functions can be modified by the user. In the event that WinCC is reinstalled or upgraded, the standard functions, that are modified, are deleted or replaced by the included standard functions. Therefore, users should save the modified functions before hand.

For the design and editing of functions and actions, WinCC includes the editor "Global Script" You start "Global Script" from the Navigation window in WinCC Explorer.

6.1.5 Project functions - characteristics

6.1.5.1 Project functions

- are created by the user
- can be modified by the user
- can be protected with a password against modification or viewing by unauthorized parties
- have no trigger
- are only known within the project
- have file names in the form of "*.fct"

6.1.5.2 Project functions can be used

- in other project functions
- in Global Script actions
- in Graphics Designer in C actions and within the Dynamic Dialog
- in Alarm Logging within the Loop in Alarm functionality
- in Tag Logging when starting and releasing archives and when backing up short-term archives.

6.1.6 Standard functions - characteristics

6.1.6.1 Standard functions

- are included in WinCC
- can also be created by the user
- can be modified by the user
- can be protected with a password against modification or viewing by unauthorized parties
- have no trigger
- are registered in all projects

- have file names in the form of "*.fct"

6.1.6.2 Standard functions can be used

- in project functions
- in other standard functions
- in Global Script actions
- in Graphics Designer in C actions and within the Dynamic Dialog
- in Alarm Logging within the Loop in Alarm functionality
- in Tag Logging when starting and releasing archives and when backing up short-term archives.

6.1.7 Internal functions - characteristics

6.1.7.1 Internal functions

- are included in WinCC
- cannot be created by the user
- cannot be modified by the user
- cannot be renamed
- have no trigger
- are registered in all projects
- have file names in the form of "*.ict"

6.1.7.2 Internal functions can be used

- in project functions
- in standard functions
- in actions
- in Graphics Designer in C actions and within the Dynamic Dialog

6.1.8 Local functions - characteristics

6.1.8.1 Local functions:

- are created by the user
- can be modified by the user

- can be protected with a password against modification or viewing by unauthorized parties
- have at least one trigger
- are only performed on the assigned computer
- have file names in the form of "*.pas"

6.1.8.2 Local functions can be used

Actions are used for picture-independent background tasks, e.g. the daily printing of a report, the monitoring of tags or the execution of calculations. The execution of an action is started by the configured trigger. Before an action can be executed, Global Script Runtime must be placed in the startup list.

In contrast to global actions, local actions can be assigned to a specific computer. In this manner it is possible to ensure, for example, that a report will be printed only on the server.

6.1.9 Global actions - characteristics

6.1.9.1 Global functions:

- are created by the user
- can be modified by the user
- can be protected with a password against modification or viewing by unauthorized parties
- have at least one trigger, which starts the execution
- will be executed in a client-server project on all of the project's computers
- have file names in the form of "*.pas"

6.1.9.2 Global functions can be used

Actions are used for background tasks, e.g., the daily printing of a report, the monitoring of tags or the execution of calculations. The execution of an action is started by the configured trigger. Before an action can be executed, Global Script Runtime must be placed in the startup list.

In contrast to local actions, global actions are performed in a client-server project on all of the project's computers. In a single station project, there is no difference between global and local actions.

6.2 Global Script Editor[1]

For the creation and editing of functions and actions, WinCC includes the editor "Global Script" Users start "Global Script" from the Project window in WinCC Explorer.

The Global Script editor satisfies the usual Windows standards. It has toolbars, a menu bar and a status bar. It has several windows, with have pull-down menus. The **Navigation window** is used to select functions and actions, to edit them or to position the text cursor in an Editing window. The functions and actions are gathered in hierarchically ordered groups. Functions are displayed with their function names, actions with their file names.

Functions and actions are written and edited in the **Editing window**. It is only displayed when a function or action has been opened for editing. Each function or action will be opened in its own Editing window. Several Editing windows can be opened simultaneously.

The **Output window** is used to display the results of the functions "Search files" or "Compile all functions". By default, it is visible but it can be hidden, if necessary.

For each compiled function, the warnings and error messages from the compiler if there are any will be displayed. In the following lines, the path and the file name of the compiled function, plus the summarizing message from the compiler, will be displayed.

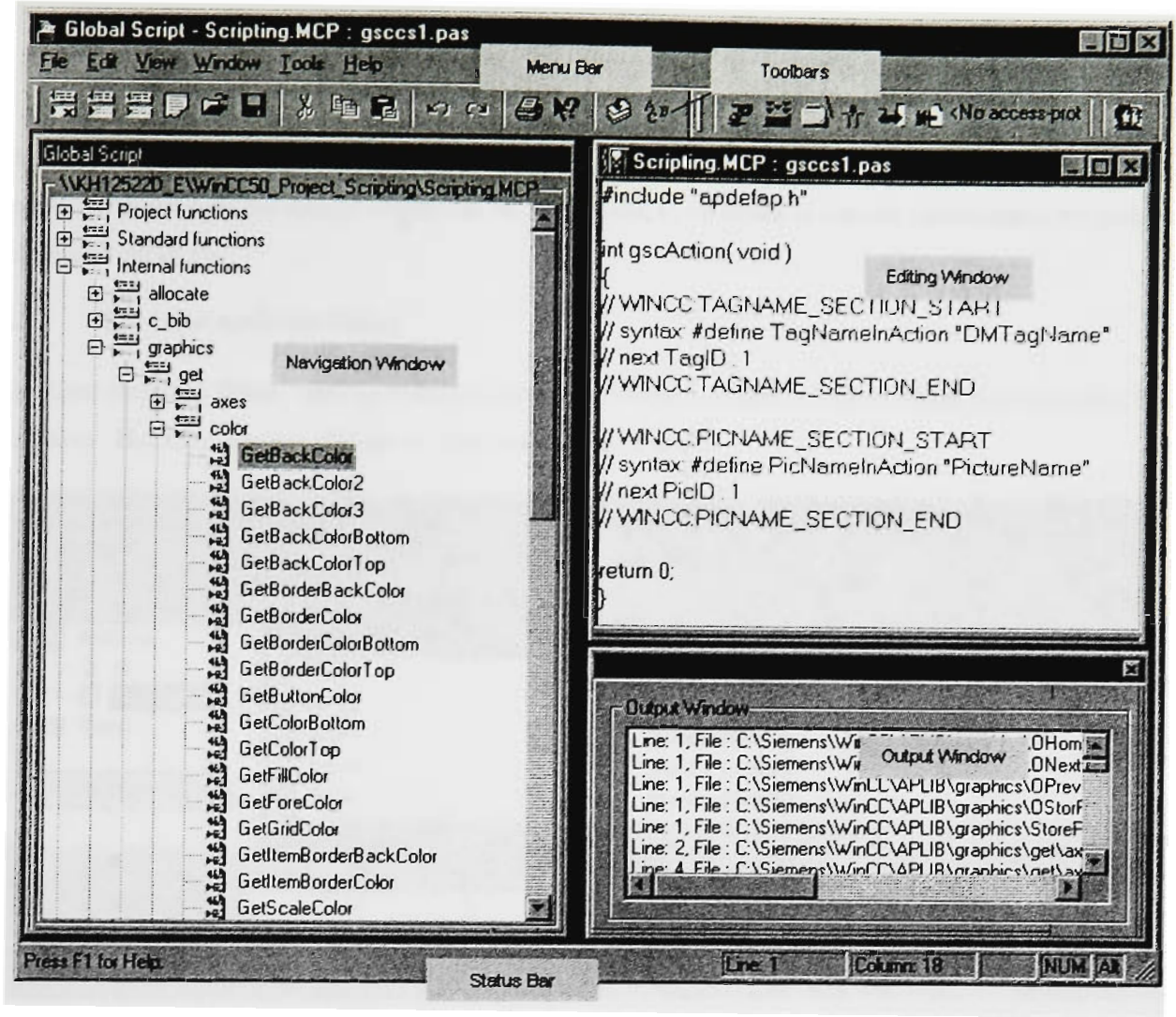


Figure 6.4: The Global Script Editor

6.3 User Archive[2]

Data from technical processes can be stored continuously on a server PC via the User Archives of WinCC. In the Graphics Designer, a WinCC User Archives Table Control can be configured to display online data from the User Archives in table form during runtime.

The WinCC User Archives offer two types of database tables:

- Archives: Archives are database tables where users can set up their own data fields. Archives store data and provide standardized access to these data following SQL database conventions.
- Views: Views receive data from the archives and group that data, e.g., to form overviews about product groups.

There are two ways to create User Archives:

- The User Archives Editor for a convenient, interactive configuration
- The User Archives Script Functions for configuring in the WinCC script language

The User Archives Script Functions also allow the implementation of various actions for the runtime operation. In the runtime screen, a table can be configured, to be which is directly connected to the process screens of the PLCs.

6.3.1 The User Archives Editor

The User Archives Editor, with its Windows-like user interface, makes it easy to set up and maintain User Archives. The User Archives Editor is separated into three areas:

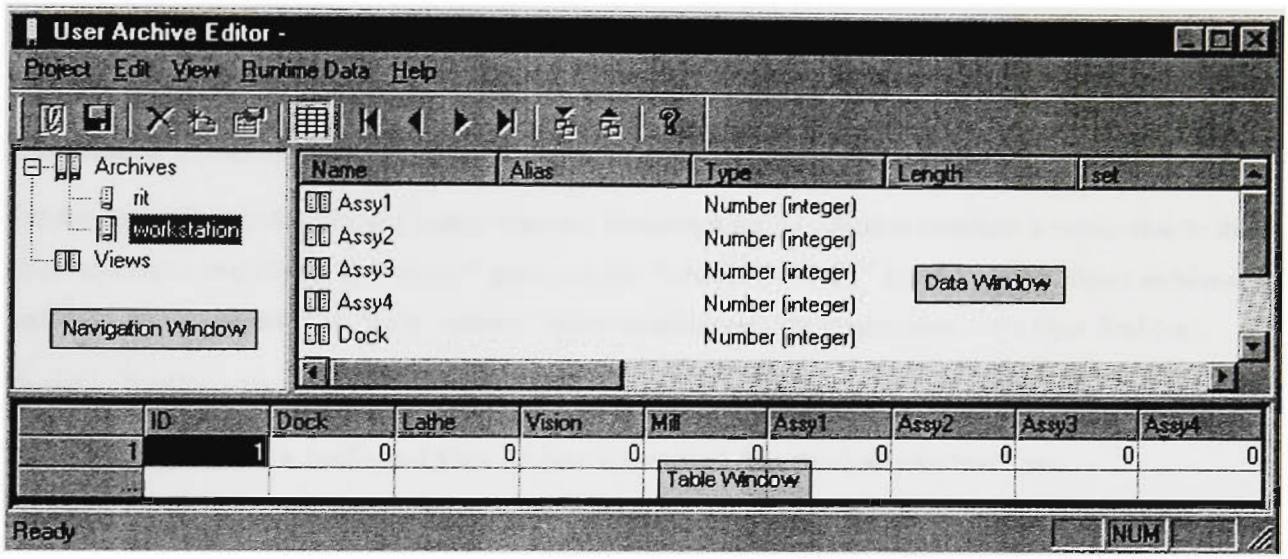


Figure 6.5: The User Archive Editor

The **Navigation Window** (the window at the top left) for selecting archives and views.

The **Data Window** (the window at the top right) for displaying and editing fields. The Data Window displays the fields of the archives and views that were selected from the navigation window.

The **Table Window** (the window at the bottom) for displaying and changing online data of the selected archives and views. In the table window of the User Archives Editor, an online connection to the process screens of the PLC can be made.

6.3.2 User Archives Script Functions

The User Archives Script Functions can be divided into

- Configuration Functions for configuring User Archives
- Runtime Functions for configuring various actions for the runtime operation

The User Archives functions are activated by actions in the runtime screen, for example a mouse click on a certain button. The WinCC script language has similarities to the high level C language, and the database functions are based on the SQL standard.

6.3.2.1 Standard Script Functions

WinCC offers a number of script functions to allow the user a flexible utilization of the User Archives. The script functions of the User Archives follow uniform name conventions. All User Archives script functions start with "ua", for example "uaConnect", "uaArchiveOpen", uaArchiveGetFields", etc. User Archives Runtime functions always start with "uaArchive"

The User Archives functions are divided into configuration and runtime functions. Handles are required for the configuration and runtime functions, which are returned by the previously called functions "uaQueryConfiguration", "uaConnect" and "uaOpen"

Establishing a Connection to the User Archives

For the access in runtime, the *uaConnect* standard function must be called to establish a connection to the User Archives component. "uaConnect" generates the "UAHCONNECT" handle, which allows archives and views to be opened. The "uaDisconnect" function terminates the connection to the User Archives.

Opening Runtime Functions

For runtime operation, a configured User Archive is required. The "uaQueryArchive" and "uaQueryArchiveByName" functions provide a handle for the runtime functions. After is opened the archive with the "uaArchiveOpen" function, the User Archives runtime functions can be used.

Functions for the Runtime Operation

The "uaArchiveNext", "uaArchivePrevious", "uaArchiveFirst" and "uaArchiveLast" functions move the pointer. A unique assignment to a data record of the User Archive is generated via the "hArchive" handle. This assignment also allows indirect addressing, as required by the screen dialog boxes.

The "uaArchiveUpdate" function stores the temporary data record in the archive and overwrites the data record to which the pointer is currently pointing. This data record must previously be read by the "uaArchiveNext", "uaArchivePrevious", "uaArchiveFirst" or "uaArchiveLast" functions.

6.3.2.2 Handles for Runtime Archive Functions

The "uaConnect" User Archives function generates the "UAHCONNECT" handle, which is required for opening and closing archives and views. This means that the "uaConnect" function must be called first in order to receive the "UAHCONNECT" handle. This handle then allows users to call the script functions listed below for opening and closing archives and views. Finally, to complete the configuration, "uaDisconnect" must be called.

The "uaQueryArchive" and "uaQueryArchiveByName" functions generate the "UAHARCHIVE" handle. This handle is required for the "uaArchiveOpen" User Archives script function, which opens the archive for the runtime operation. To terminate the connection, the "uaRelease" and "uaArchiveClose" functions must be called. Refer to Appendix E for a list of Handles for Runtime Archive Functions.

6.4 Open database[1]

Two databases are required for each WinCC project: a Configuration database (CS database) with the statistical configuration data and a Runtime database (RT database), which contains the process data. Users create the project data during calibration; WinCC creates the process data during the operating process. Database technology from Sybase SQL Anywhere is employed for both databases.

The MES and ERP applications can use the database query language SQL or ODBC drivers to access the process values. The data can, for example, also be imported into an ORACLE database using the Sybase Connectivity Tools.

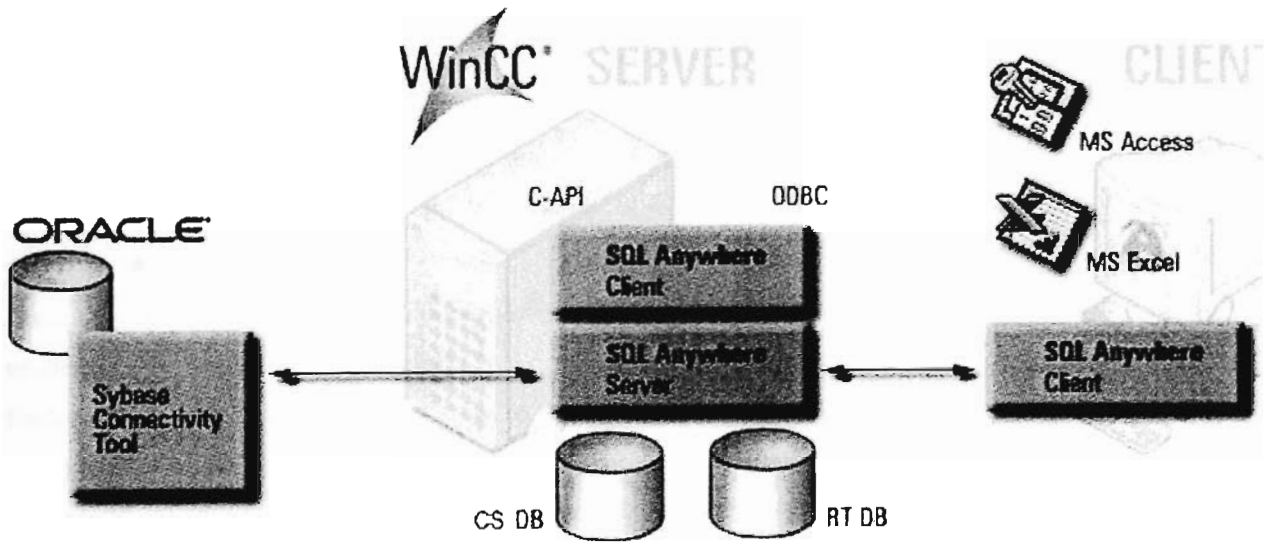


Figure 6.6: WinCC Open Database

Reference:

[1]Siemens (2001,December) "Simatic HMI WinCC 5.1" WinCC Information systems.

[2]Siemens (1999, August) "Simatic HMI Options" manual.

Section 7. Description of the CAMCELL

The CAMCELL is an integrated manufacturing facility that has evolved over the past several years within the department of Industrial and Manufacturing Engineering at Rochester Institute of Technology. It is a completely automated and fully operational production facility.[2] In order to completely understand the various aspects of the CAMCELL, it is described from the following four perspectives in this section:

1. Manufacturing and Material Handling
2. Product/Process Flow
3. Computer Hardware Architecture
4. Software Architecture and Information Flow

7.1 Manufacturing and Material Handling[2]

The facility occupies about 600 sq.ft. in area and is comprised of two CNC machining centers (a mill and a lathe), a vision/inspection station and a load/unload dock. These four stations and four assembly stations are connected by two-closed loop conveyor systems and are the eight main functional stations in the facility. The two conveyors are placed end to end in an "L" shape configuration.

The main conveyor loop connects the four stations identified above while the secondary conveyor, which is perpendicular to the main loop, connects the assembly stations. These conveyors act as a buffer and material transport system for up to twenty 8 in. x 8 in. size pallets, that hold the fixtures and materials needed in the CAMCELL operations.

Each pallet carries a Radio Frequency tag for identification that is tracked by two NAMCO radio frequency identification systems placed on the two conveyors. Each tag has capability to write 16 bytes of data. The tags can be re-written in order to maintain additional work in process (WIP) status.

An industrial grade robot provides material handling for the mill center and the vision station. A robot and an elevator mechanism developed in house provide the load/unload capability at the lathe center. In addition to this, one Intellidex robot, one IBM robot and two Adept robots are used at the four assembly stations.

In summary, the following is a list of primary manufacturing and material handling equipment in the facility:

- Two 10-foot long BOSCH conveyors
- One 3-axis TERCO CNC vertical mill
- One 19-inch ORAC CNC lathe with 8 station turret
- An IRI vision inspection system
- A 2-axis elevator
- A 3-axis pneumatic robot (manipulator)
- A NAMCO radio frequency identification system
- Two 5-axis Intellidex robots

- Two ADEPT robots
- One IBM robot

7.2 Product/Process Flow[2]

As explained in the previous perspective, the two major pieces of production equipments in the facility are the two machining centers: a CNC lathe and a CNC Mill. Thus, the facility is capable of producing any type of parts that can be manufactured by a lathe or a mill or a combination of those two. The Primary material used in manufacturing the parts is a special wax like material. However, the equipment is also capable of machining metal parts. Some of the sample parts that are produced in the facility are screws, nuts, and a RIT key-chains. The facility illustrates the operations of a flexible manufacturing system (FMS) in the scheduling of machines, assignment of pallets and flexibility in making any parts in any order with necessary setups and machine programming handled automatically by the CAMCELL control software. The materials and the tooling fixtures for parts being produced are loaded at the dock. These will travel to the lathe and/or mill centers for machining and the vision station for inspection in order dictated by the process steps. All finished parts return to the dock for unloading. For example, if the part to be manufactured is a screw, then a cylindrical material will be first loaded at the dock. Then it is routed to the lathe where the threads of the screw will be cut. Next the part is routed to the mill center where a slot is cut. Finally the part is routed back to the dock for unloading. Thus, this is a typical material flow for the manufacture of the part. All equipment and the material transport system in the facility are fully integrated into a completely computer controlled manufacturing facility.

7.3 Computer Hardware Architecture[2]

As defined by the National Institute of Standards and Technology (NIST), there is a five-level hierarchy in manufacturing automation.[1] RIT's CAMCELL computer hardware architecture has evolved into one that already implements such a hierarchy. Most of the manufacturing equipment and related computer hardware used in the facility were acquired based on their individual merits for instructional purposes outside the CAMCELL environment.

The compatibility among systems for total integration into the CAMCELL is accomplished through many in-house developed hardware interfaces and control software. The five-level computer hardware architecture in the CAMCELL includes following five levels: tool, station, cell, center and plant levels. At the **tool level**, several data acquisition systems are used to monitor and manipulate the status of various elements in the system such as the latches and sensors. At the **station level**, there are several machining centers, robots and vision systems that are primarily involved in the manufacturing of parts. The main activity at the **cell level** is routing jobs through the system for all low level stations. This deals with information and material handling in order to deliver the right parts and material to the right place at the right time.

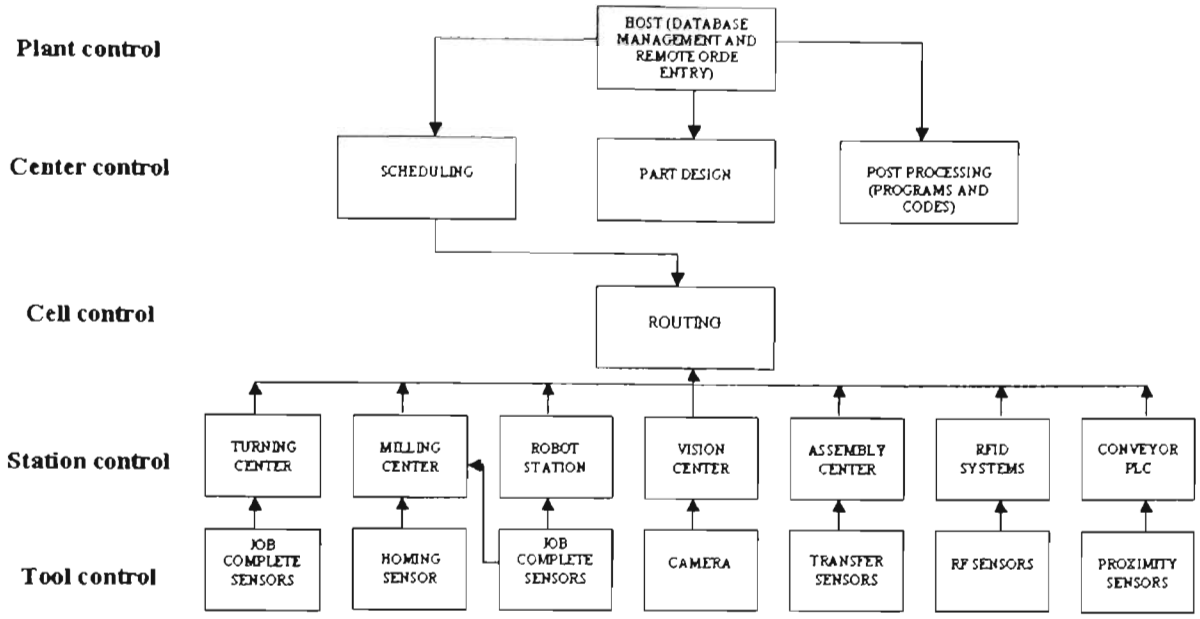


Figure 7.1 CAMCELL Hardware Architecture

At the **center level,** scheduling of orders takes place. This layer looks into job priorities, machine availability, personnel availability, and material availability as well as machine capacity limits. At the uppermost level, the **plant level,** orders from customers are entered. Other tasks performed by this level are parts catalog management, initiation of design work, price negotiations, generation of material acquisitions, inventory management, etc.

The low-level station control and device manipulation are achieved via Programmable Logic Controllers (PLCs), Data Acquisition Systems and computers. The lathe and the mill, during their machining cycle, operate under their own CNC controllers. However, their manufacturers originally designed these machines as stand-alone units for manual keyboard control. They have been modified in the CAMCELL to allow remote computer controlled operation for automatic downloading of NC codes and device manipulation, both of which are essential to integration of the mill and lathe machines into a flexible and automated manufacturing environment. The load/unload manipulator of the lathe has a dedicated computer for control. All the robots involved in the assembly operations have a controller of their own. The Intellidex robot has a programmable controller of its own. The IRI vision system is operated using a UNIX based microcomputer system. The NAMCO Radio Frequency Identification System is a host programmable intelligent peripheral. The results are delivered from the scheduler to the router on a once-a-day or once-a-shift basis, whereas the station level and tool level controllers have a real-time interaction between each other.

The integration of manufacturing systems can range from a relatively self contained machining cell to large automated factories that include support functions such as planning and scheduling. The type of architecture or combination of architectures used depends a great deal upon the application. Many of the CIM systems in use today depend upon a combination of networked and hierarchical architectures. A hierarchical design is used in the CAMCELL.

The hierarchical design architecture consists of varying levels of controls based upon the communications paths established between computers to form a control structure in the shape of a pyramid. At each level within the control structure, responsibilities are designated based upon the task to be performed and the time dependencies of that task. In this way the lowest level within the hierarchy, machine controllers or PLCs, is able to directly interact with the machines, which requires response time in the range of milliseconds. Higher-level computers are then able to co-ordinate a group of machine controllers in real-time. In this arrangement, the higher-level computer is directly linked to each of the lower level computers, sending out control messages to them and receiving some form of acknowledgement from them upon completion of the delegated tasks.

The hierarchical systems can be configured with powerful, inexpensive computers and, hence, are extremely cost-effective. A networked architecture may be used to connect various manufacturing shop floor controllers in a larger facility, with the engineering planning functions, scheduling functions and business costing programs completing the idealized CIM system. The advantages of the hierarchical system are its quick response time, low cost, and high reliability because it is are not subject to unknown process sequence. Reliability can be designed into these systems at key points in the control structure. Hierarchical architectures are ideally suited for cell control activities and as such are found more frequently within complex flexible manufacturing systems.

7.4 Software Architecture and Information Flow

The nerve system of any advanced manufacturing system is computer software and data communication. The software implements the necessary logic and intelligence in performing the required functions at the right time and in the proper sequence.[1]

The architecture of CAMCELL maps very nicely into the MES functions that were discussed in section 2. This section focuses on software architecture and the information flow that facilitate the integration of various elements in the CAMCELL system. Data flow diagrams are used to describe the software architecture that evolved from a hierarchal control system for cell controllers.

The control software is described here as five logical modules:

1. Order Manipulation.
2. Scheduling.
3. Routing.
4. Station Controller.
5. Support.

Figure 7.2 & 7.3 depict the software modules in a hierarchy dictated by data flow.

7.4.1 Order Manipulation

The ORDPRO program performs the function of receiving, validating, processing and acknowledging (accept/reject/completion) the customer orders. This function operates in the Sales Department of an enterprise.

The customer places the order through the order entry interface screen, which can be a form, designed in Visual FoxPro accessible via a Web browser. First, the customer selects the part name that she or he is interested in. Once the user selects the part name, she or he gets the necessary information about that part. This information is provided to him on the screen, from the "*Part Catalogue*" database table. This table contains the information about the part name, cost of the part and the lead-time required to manufacture the part.

After getting the necessary information about the part, the user enters the customer name, quantity desired and the required delivery date.

The ORDPRO program verifies the information received from the user and acknowledges the acceptance of the order. Then it generates an order number for the order and adds the customer name, part name and quantity required to the "*Customer Orders*" database table. The order table resides in the Enterprise Resource Planning (ERP) database and can be accessed by the engineering department for scheduling new work orders. The ORDPRO task is also responsible for handling the cancellation and the user inquiries.

7.4.2 Scheduling

The Scheduler task simulates the task of a Systems Scheduler/Manager. The Scheduler automatically schedules the orders in the system for production and assembly. It periodically scans for the new orders from the "*Customer Orders*" table that resides in the ERP server.

It checks for the availability of the required resources. The status of all the resources (machine, raw material and pallet) is available in the *"Resource availability"* table. The Scheduler program gets the checklist of all the resources for a particular work order by querying the process planning data (.SEQ file). For instance, if the required raw material is not available the Scheduler notifies to the purchasing department. When all criteria have been met, a production and sent order is initiated to the *"Scheduled Order"* table that is available for the ROUTER program.

Based on the required delivery date provided by the customer, the Scheduler assigns a priority to the order. The scheduler is now ready to schedule the next order in queue and adds the new order in the *"Customer Order"* table based on its priority. Once the order has been scheduled, the Scheduler program updates the status of an order in the *"Customer Order"* table.

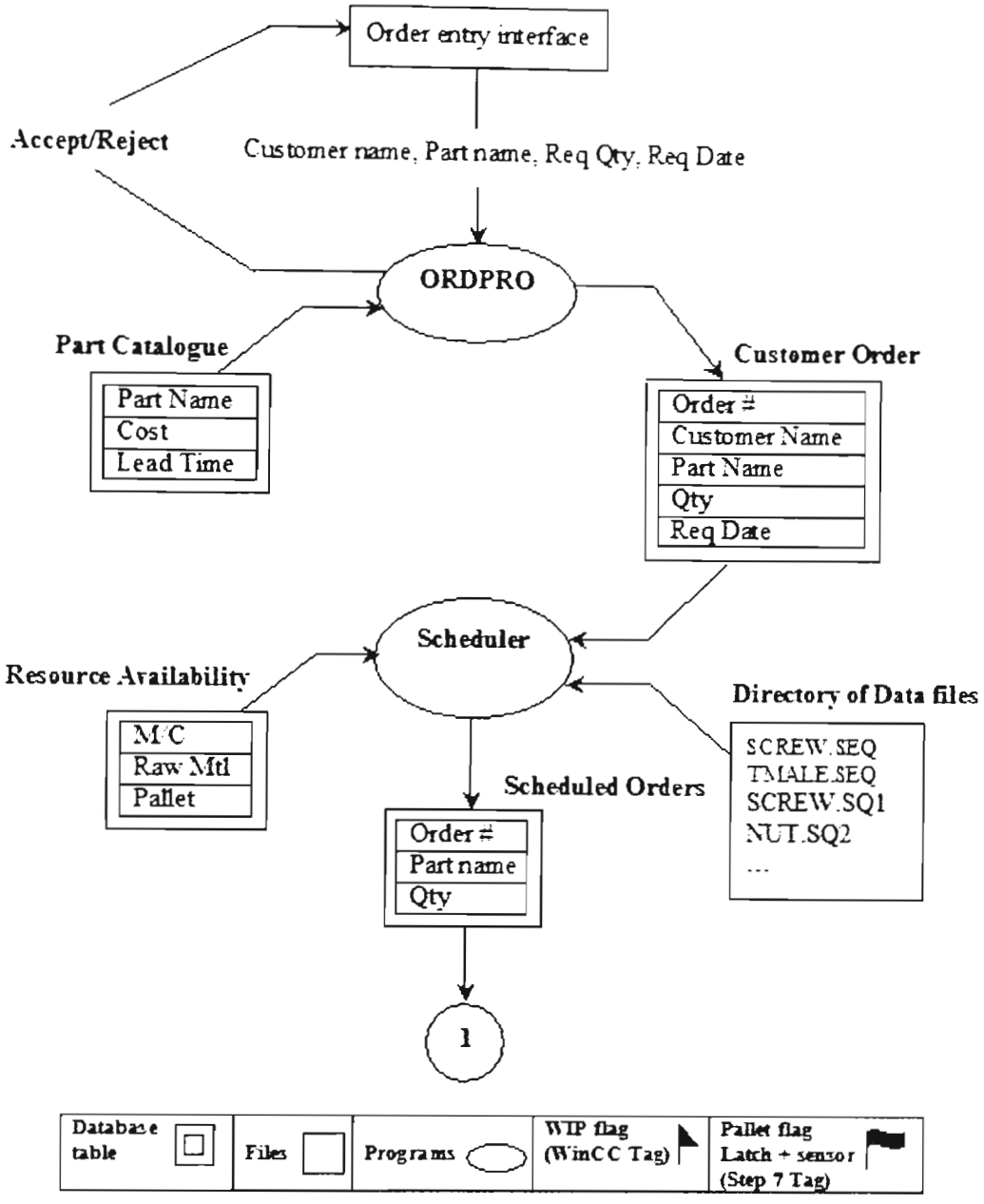


Figure 7.2 Data Flow Diagram - I

7.4.3 Routing

The ROUTER (Process Sequencer) program simulates the function of a factory floor foreman and is responsible for sequencing all the process related steps within the CAMCELL. It resides in the Manufacturing Execution System (MES) server. It is responsible for optimally selecting pallets with materials and fixtures and routing them to the individual stations as and when needed. Each of the eight stations has a primary task that interfaces with ROUTER task.

The ROUTER uses the information provided by the *"Scheduled Order"* table. It opens the next order in queue, and it looks for and opens the .SEQ file from the data directory according to the name of the part to

be produced. The .SEQ file is an overall process sequence file that contains the order of stations into which a part in its various forms will visit.

The ROUTER program checks the status of the machine by checking the **WIP Flag**  .The WIP Flag is a WinCC Tag that is a memory word data type. The Flag is set high (by ROUTER) whenever a machine is busy processing and is set low (by the station controller) when the machine is idle or completes its order. Thus the ROUTER knows the status of all workstations in the system. It interacts with the station's controller program to initiate the machining or assembly cycle by sending a message (with part name) to the station's primary task.

The ROUTER program has control over the quantity produced; it checks the quantity to be produced and accordingly sends those many numbers of requests to the workstation. It maintains the WIP information into the *"Job status"* database table. The *"Job status"* table contains information such as Order Number, Part Name, Total Ordered quantity, Completed quantity, Rejected quantity, WIP and the Job location. When an order is in process, the WIP field of *"Job status"* table is set to "1" by the **WIP Flag** and the Job location is set to the name of the station where the job is currently being processed. This information provides the database clients real-time status of the job and workstation. It can be further manipulated to obtain the aggregate data that will be fed back to the ERP database for corporate and customer inquiries. After a work order is completed, the record is deleted from the ""*Scheduled Order*" table. An event history is also generated that provides information on each order's starting and finishing times.

7.4.4 Station controller

Each of the eight stations has a primary task (LATHE, MILL, VISION, DOCK, ASSEMBLY1, ASSEMBLY2, ASSEMBLY3, ASSEMBLY4) and one or more secondary tasks. The purpose of the primary task is to receive the machining or assembly request from the ROUTER, to execute the necessary steps (such as requesting required material and fixtures and downloading the NC code to machine and robots) as instructed in the station specific sequence file (.SQX) , and finally to notify the ROUTER. Each of the NC machines and robots has its own task of reporting to the station's primary task. The LATHE task communicates with the MANIPULATOR task for machine loading and unloading operations. The MANIPULATOR task controls the robot that actually performs the load/unload functions. The MILL task interacts with the ROBOT task, which coordinates the use of an Intellidex robot between mill and vision stations. MILL also communicates with the TERCO task that controls the Terco CNC milling machine. The vision task also interacts with the ROBOT task and is responsible for inspection of parts.[1]

As mentioned in the previous section, the ROUTER sends order information to individual stations programs. The station programs are developed on the WinCC platform. These programs open the .SQX file that has the sequence describing the process planning steps within the station for that particular part. The .SQX file resides in the library of data files and each station control program can access the file by the

part name. The **WIP flag** is reset by the individual Station programs, thereby informing the ROUTER about the order process completion.

The workstations request a pallet type from "Pallet Controller" to load or unload a semi-finished or finished part. The "Pallet Controller" program is a Step 7 program that sends the pallet request from the workstations to the "Inquire" program (which keeps the track of the pallets). The workstation sends the required pallet number in the form of a Step 7 Tag to the "Pallet Controller" program, which in this case is named the **Pallet Flag**  This **Pallet Flag** is set high when the station controller requests it. Once the pallet arrives at the station that requested it and the transfer of material is complete, the **Pallet Flag** is reset.

7.4.5 Support functions[2]

The two support functions, Inquire and Pallet controller Inquire Program, are described below.

7.4.5.1 Inquire program

The Inquire module has the task of monitoring the movement of the pallets across the eight stations, monitoring the status of the latches at the eight stations, and providing the information about the next arriving pallet at each of the eight stations. The Inquire module also has the task of stopping or releasing a pallet at a particular station, when requested by the Pallet Controller. The Control program to implement these functions is built using the Siemens' Step7 Ladder Logic programming language.

7.4.5.2 Pallet Controller

The Pallet Controller module passes on the pallet request information to the Inquire module, which on the basis of the information about the next arriving pallet at the station, stops or releases the pallet at that station.

All the information used in the CAMCELL control is organized in the overall data management system as described in the next chapter.

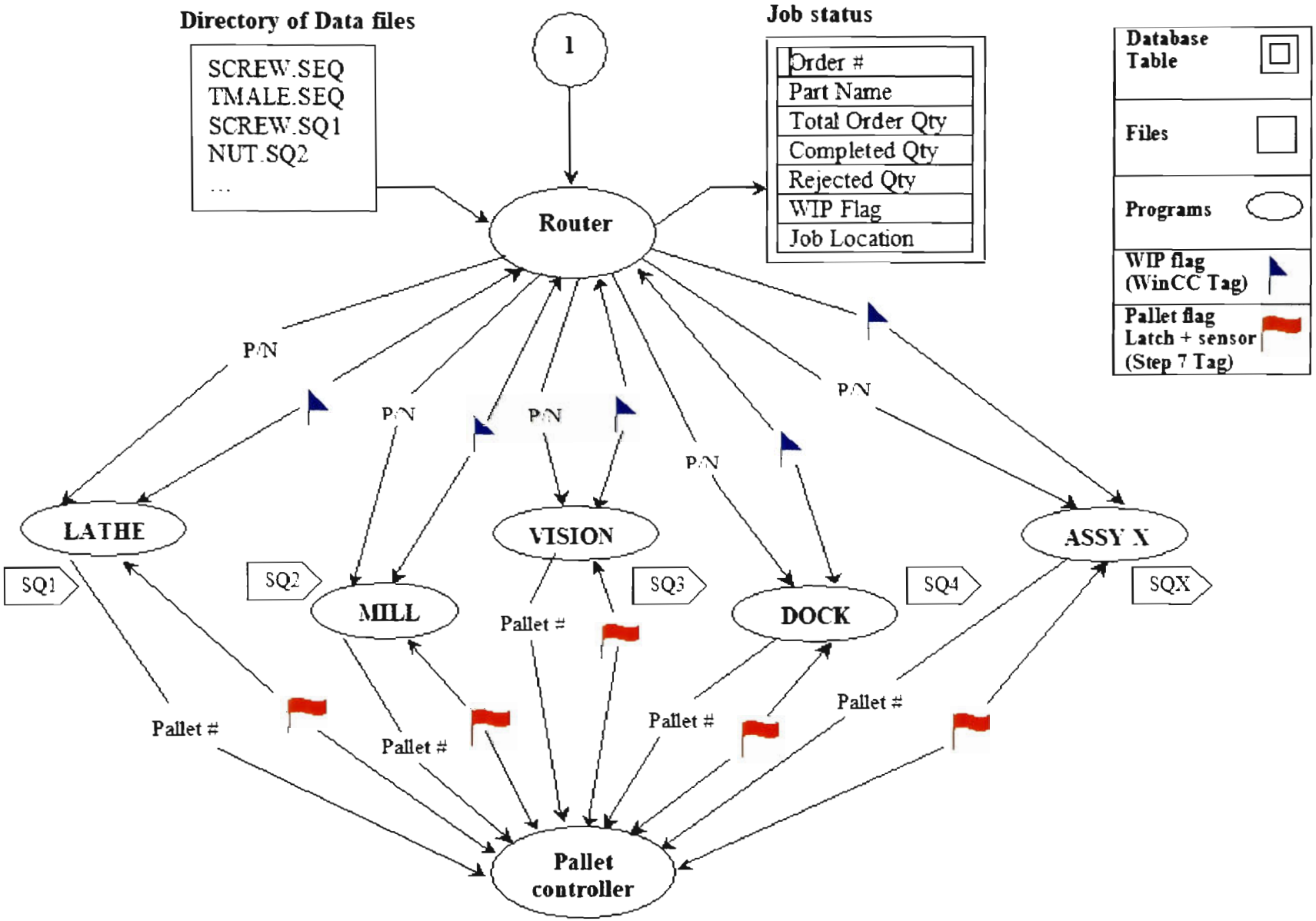


Figure 7.3 Data Flow Diagram - II

Reference:

[1]Paidy, S.R and Reeve, R, "Software Architecture for a Cell Controller"

[2]Palsule A, and Paidy. S.R. (2002, August)"System integration using Siemens PC based automation Technology"

Section 8. A CIM database for Real-time data

As discussed in the previous chapters the problem with ERP systems is their blindness to activities on the shop floor and the timeliness of data. MES provides an enterprise-wide, real-time view of the complete manufacturing environment. It closes the gap between the ERP and production worlds so that optimal execution is realized.

The aim of this study is to develop software architecture for the CAMCELL application in order to demonstrate vertical integration, i.e., connection of the ERP with the process control systems of a manufacturing system. The connection is achieved by implementing a Manufacturing Execution System (MES). Similar to the CAMCELL Factory described below, a production plant using this technology should be equipped with Simatic Step-7 as the control system and WinCC as the monitoring system.

8.1 Data Flow from ERP to MES

The CAMCELL Factory consists of various departments similar to any conventional manufacturing organization. We have sales, engineering, purchasing, finance and manufacturing linked to a corporate level system called an ERP server (see Figure 8.1). The Sales department interacts with customers and takes their orders. These orders are placed through the order entry interface screen, which can be a form, designed in Visual FoxPro [insert a space] (VFP) and accessible via a Web browser. A form similar to that can be used for order inquiry and order cancellation. The order information is collected in the Sales department database, which could be a Microsoft SQL server. The Sales department updates the ERP server with the new orders.

The production planning is carried out by the Engineering department, which schedules the new orders and checks for the availability of the required resources, such as raw materials, tools, pallets and machines. The data related to the availability of the raw materials, tools and pallets are available in the inventory database that is managed by the Purchasing department. Purchasing is then notified about the non-availability of any resources. The SCHEDULER prioritizes the orders according to the required delivery dates.

The functions of ordering and scheduling are performed in the ORDPRO and SCHEDULER modules, which are discussed in detail in the previous chapter: "Software Architecture and Information Flow"
Restrictions are applied depending on who accesses the data from the ERP server.
An MS SQL Data Transformation Service (DTS) package could be scheduled at the beginning of each day that would update the MES database for the new orders that the ERP scheduler has scheduled.

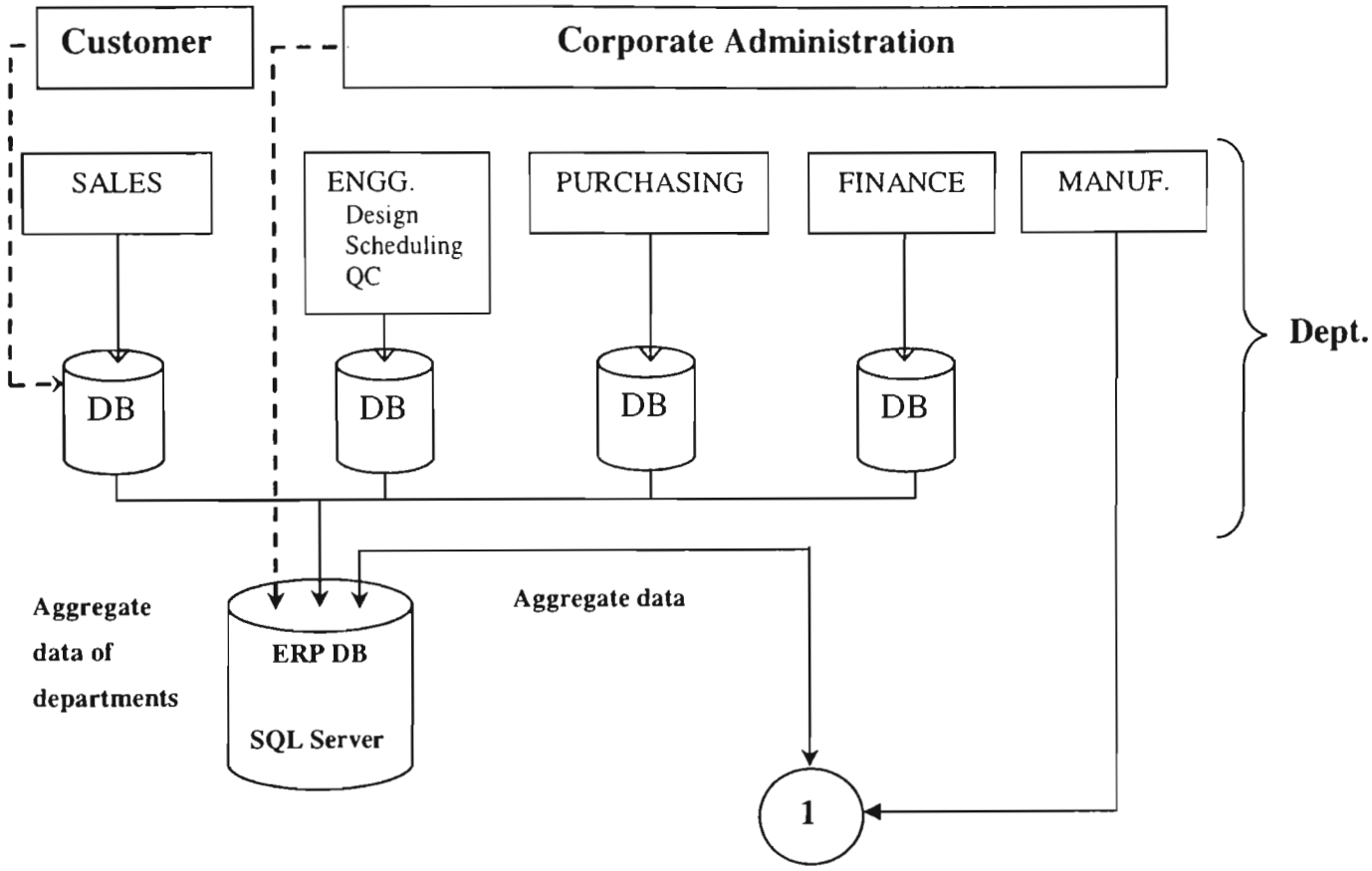


Figure 8.1 ERP Layer of CAMCELL

8.2 Data flow between MES and Control layer

The MES layer executes the plan developed by the scheduler. It performs the following functions:

- Checks for new orders in the system
- Performs the routing functions.
- Updates the job status at the end of each day.

The MES Database and a Router WINCC program perform these functions. The MES database is logically divided into two databases, one for the real-time process data (dynamic database) that it co-ordinates with the shop floor, and the other for the aggregate data (static database) that is fed from and updated to the ERP layer (see Figure 8.2). These two parts are named "CELL MES" and "ERP MES" respectively. The MES database below is a Visual FoxPro database.

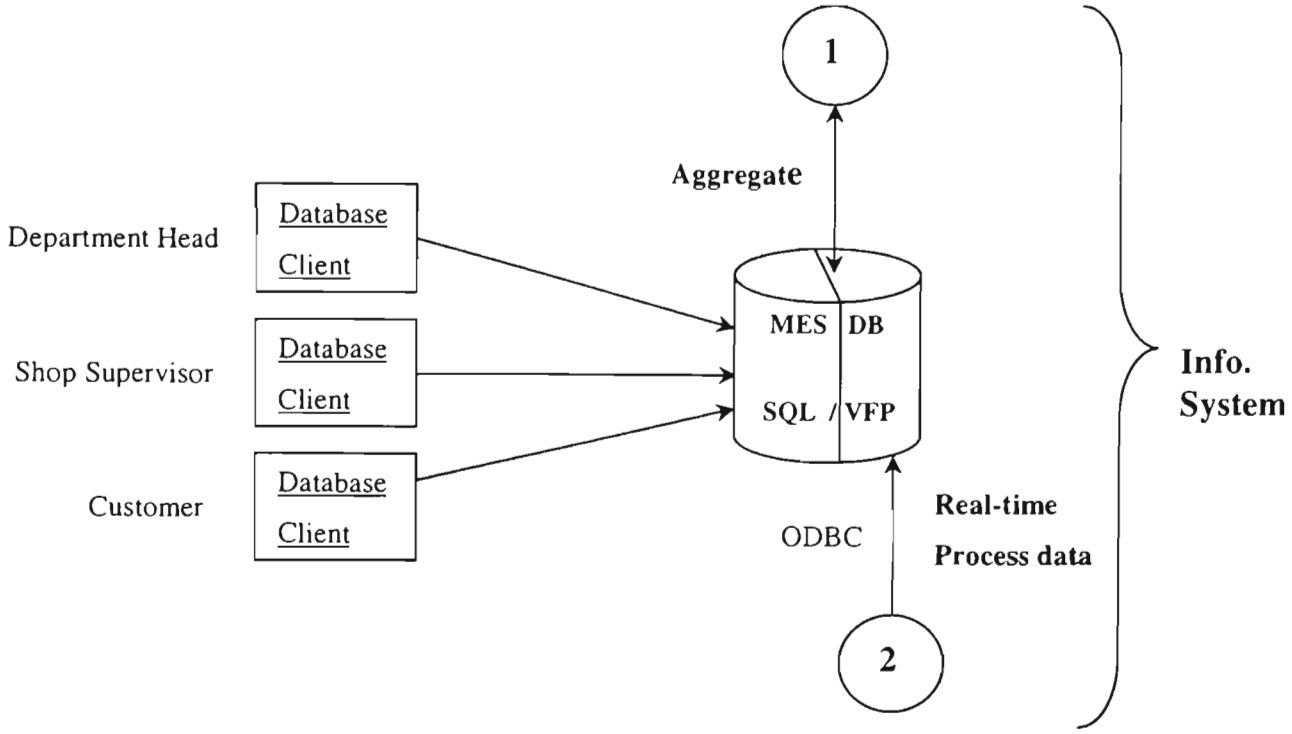


Figure 8.2 MES Layer of CAMCELL

The "ROUTER", which is a WinCC program and is explained in detail in the chapter "Software Architecture and Information Flow", performs the function of an MES. It looks for the new orders in the ERP database at the start of every day and adds it to the "ERP MES" database table. This table contains the new orders as well as the pending orders. The "ROUTER" processes the orders according the priority assigned by the "SCHEDULER" The overall process sequence file (.SEQ) contains the order of stations in which a part in its various forms will visit. A sample of an SEQ file is shown in Figure 8.3 while Figure 8.4 describes the typical format of an SEQ file. Appendix F gives the file format of a .SQX file. The ROUTER reads the .SEQ file from the directory of data files for that particular part. It transforms the order data and part names, into WinCC tags that are fed to the workstation controller program, which is also a WinCC program (e.g., LATHE Controller, MILL Controller, VISION Controller, etc.) Once the workstation program gets the part name, it begins to process the part. It also requests the inquiry program to make the required pallet type available at the station.

UL 4 3 1 SCREW

LD 1 3 1 SCREW

LD 1 3 5 SCREW

UL 1 3 6 SCREW

LD 2 3 6 SCREW

UL 2 3 7 SCREW

LD 4 3 7 SCREW

Figure 8.3 SCREW.SEQ

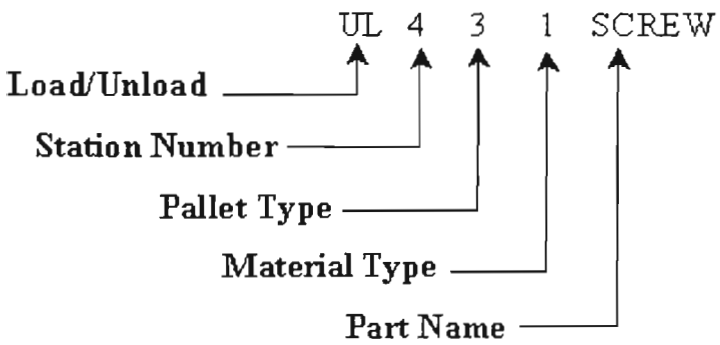


Figure 8.4 SEQ File Format

Pallet types

1 = Generic
2 = Square Stock
3 = Cylindrical stock (SCREW)
4 = Cylindrical stock (SHELL)

Material Types

1 = Raw material screw
2 = Raw material shell
3 = Raw material tmale/tfemal
4 = Raw material cube
5 = empty collar
6 = collar w/screw (no slot)
7 = collar w/screw (finish)
8 = finished tmale
9 = finished tfemale
10 = finished shell
11 = finished cube
12 = shell w/out milled holes
13 = sensor
14 = raw material nut
15 = finished nut
16 = finished screw w/out collar
17 = finished screw set

Figure 8.5 SEQ File Topology

The OPC Client-Server methodology is used to share the tags across the various functional modules. The computer to which the Siemens PLC is connected is configured as the OPC Server. All the functional modules that run on separate computers are configured as OPC Clients. These client computers access the tags resident on the OPC server via the OPC channel. In this OPC Client-Server architecture, the logical data flow differs from the physical data flow between the functional modules within the Control layer. For example, there is a direct logical data flow between the Router and the Lathe Controller. However, physically the data flow between these two modules is via the OPC server. Refer to Figure 8.6

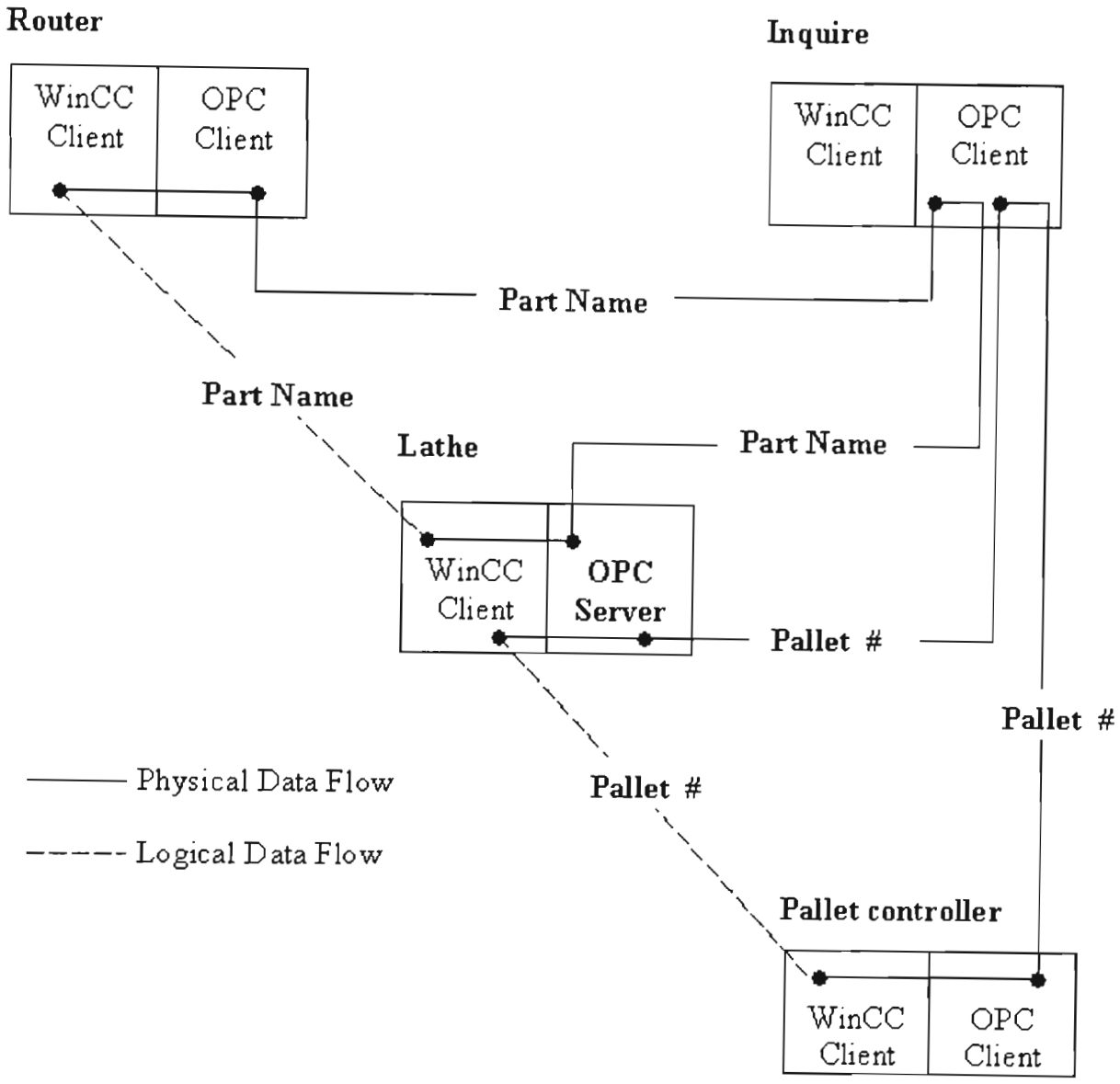


Figure 8.6 Dataflow between the MES and control layer

As described in earlier section WinCC has capability of running C scripts that are written in the Global Script editor. The Global Script has standard functions, which does the tag transformation i.e., setting the value to a tag and getting the value from the tag.

8.2.1 Data flow from Visual FoxPro (VFP) Database to WinCC

In order to read the data, example part name, from VFP's "*order_data*" database table Microsoft's ADO (ActiveX Data Object) concept is used. The intent of this work is to demonstrate the communication between the database and WinCC tag. (Refer Appendix A for step by step instructions to read data from Visual FoxPro database and write it to WinCC tags). A WinCC Screen

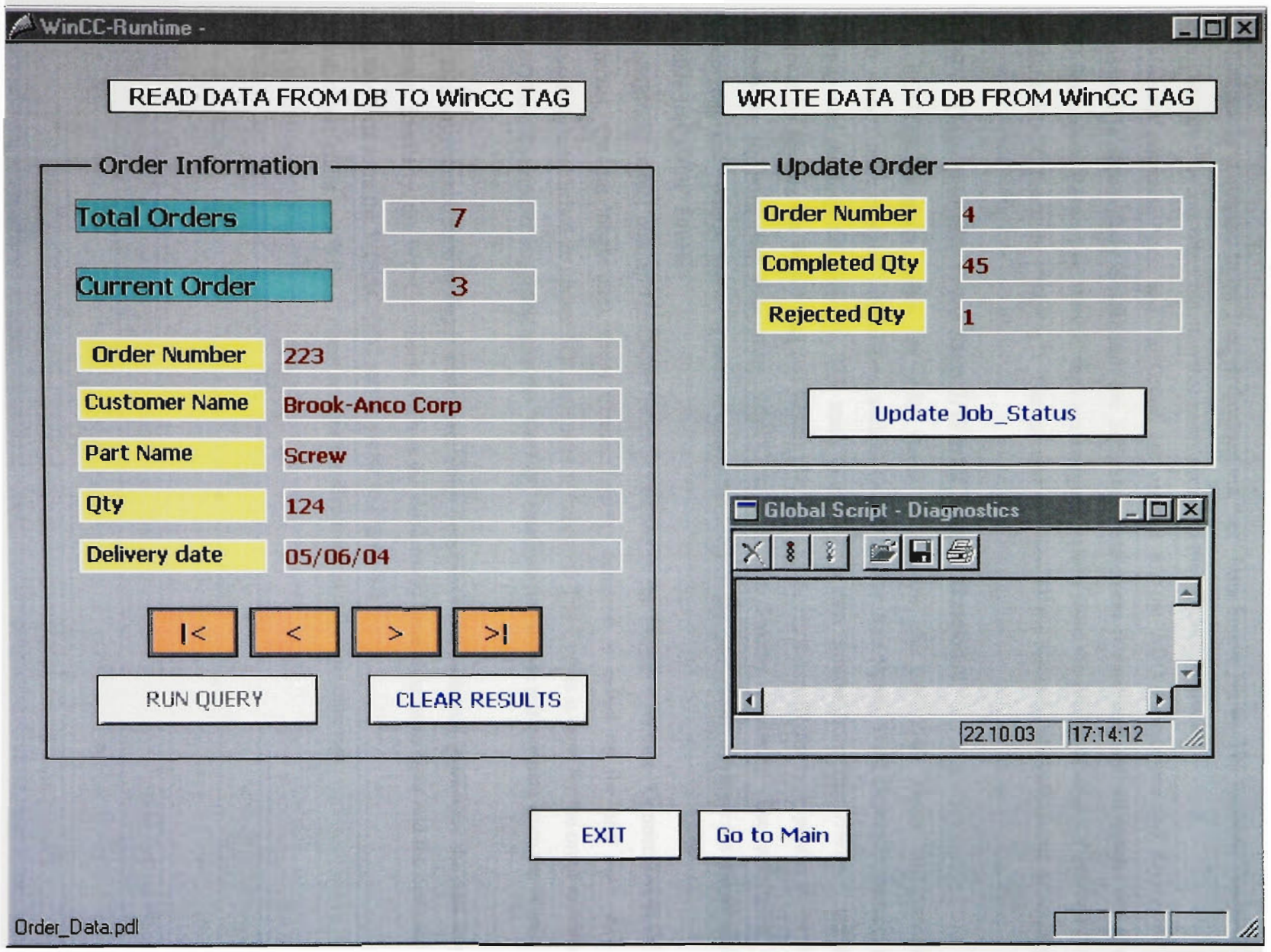


Figure 8.7 Order_Data WinCC screen

"Order_data.pdl" is developed (refer to Figure 8.7) to display Order Information that will be retrieved from "*order_data*" table of "Order" Database. On the mouse-click event of the "RUN QUERY" button, a C code is executed that reads data from the FoxPro database and writes to the WinCC tags.

To do so, a Project Function "ConnectToDB.fct" is written in the Global Script editor of WinCC. The purpose of this function is to connect to the FoxPro database via "set" Data Source Name. The function ConnectToDB() can be can be called by any other function that needs access to the "Order" database.

Two object pointers, *gcn and *grsOrder, are created for the ADO Connection and Recordset. Once the connection is successfully established, the SQL select statement is executed and the results are stored in the grsOrder object. Respective WinCC tags are assigned values from the recordset object ("grsOrder") using the standard WinCC functions ("SetTag"). The operator can view all the orders by clicking on the navigation buttons.

8.2.2 Data transfer from Step 7/WinCC tag to Database

The ROUTER program writes the order status information into the "*Job_ status*" database table. The "*Job_status*" table contains information, such as Order number, Part Name, Total Ordered Quantity, Completed quantity and the Rejected quantity. This study demonstrates data transfer from Step 7/WinCC tag to the VFP database by reading the WinCC tag and writing the value to the database. (Refer to Appendix B for step by step instructions to write the data from WinCC tags to the FoxPro database). The previously developed "Order_data.pdl" WinCC screen is used to update the order in the "*job_status*" table on the mouse-click event of "Update Job_status" button.

An additional object pointer *grsOrderUpdate is defined in the Project Function "ConnectToDB.fct" for ADO Recordset. On the mouse-click event ConnectToDB() function is called on the and once the connection successfully established an update SQL statement is executed. The Update is done for the order number entered in the I/O box. The update statement uses the "GetTag" standard function to retrieve data from the WinCC tags.

The Database clients, such as department head, Shop Supervisor and customer, get/access the data from the MES database. Similarly, the WINCC clients, such as the department head shop supervisor and the operator, get/access real time data form the WINAC OPC server. Refer to Figure 8.8

The database client and WINCC client screens are discussed in following chapters.

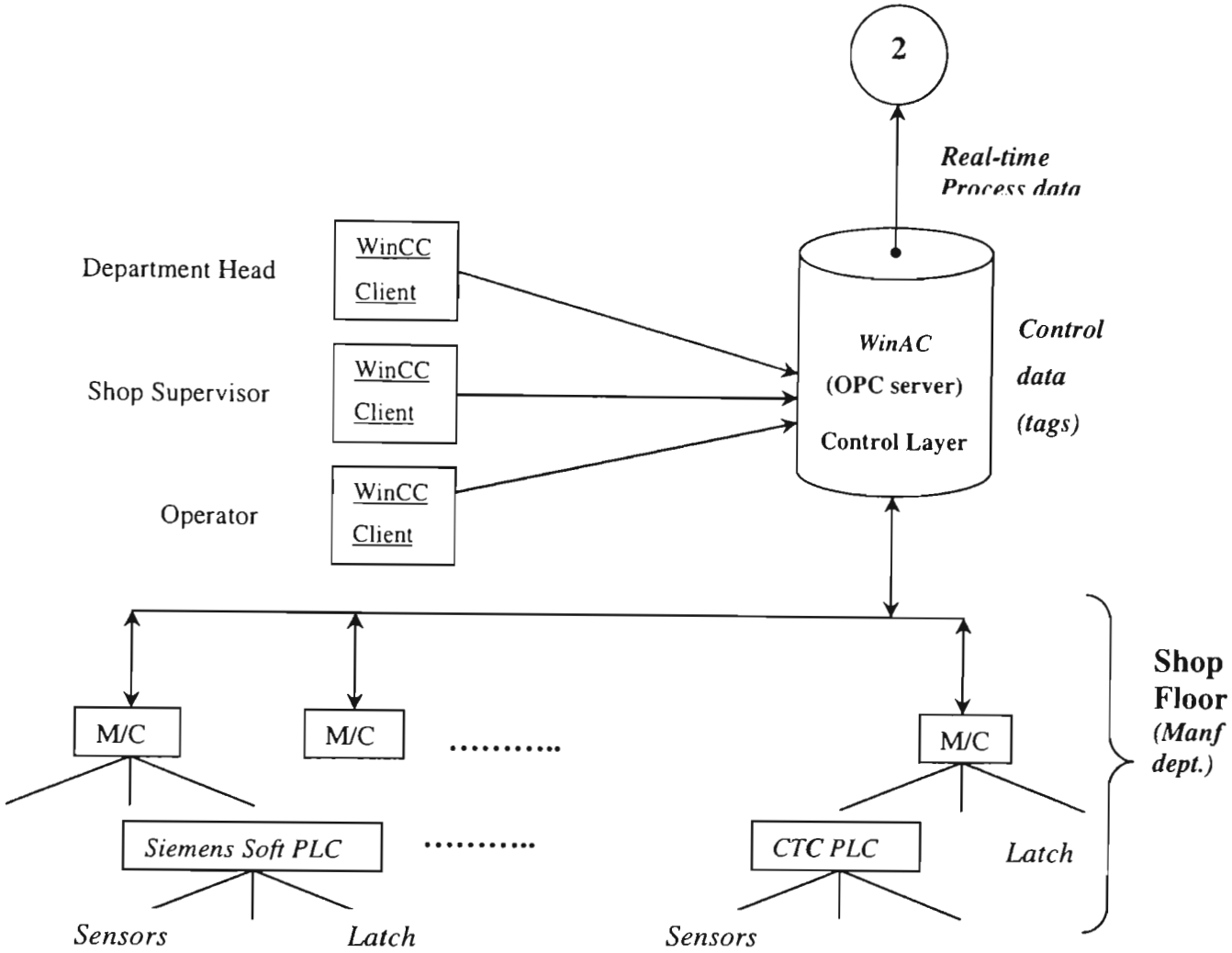


Fig 8.8 Control Layer of the CAMCELL

8.2.3 Real Time Data capture from WinCC into FoxPro Database

MES takes the plant-wise manufacturing decisions. It provides an enterprise-wide, real-time view of the complete manufacturing environment by working with time factors of 10x. The important part of this study is to demonstrate real-time data capturing from the field devices of the CAMCELL system. The MES uses "Inquire" and "Pallet control" Step 7 programs and "Camcell_OS" WinCC application already created for the CAMCELL system.

Real-time data is captured for Pallet count at every station in a FoxPro database, "RealTimeDB. (Refer to Appendix C for step-by-step instructions to capture Real-time Step 7 & WinCC data into the FoxPro database).

A new Archive "workstation" is created in the User Archive Editor of the WinCC system. The user archive creates a new tag group "@UA_workstation" in Internal tags of the Tag Management. New fields are created in the "workstation" User Archive; these fields are User Archive WinCC tags that correspond to Step 7 tags, which in turn represent the field device. The UA WinCC tag continuously scans the Step 7 tag to get the real-time data. This is achieved by the **Global Action** and **Trigger** functions of the Global Script Editor. Actions triggering anything on a Step 7 tag changes its value. The action is a C script to set the value of the Step 7 tag to the UA WinCC tag.

WinCC uses the "Sybase Adaptive Server Anywhere" database to store all the Run-time process values. Sybase contains the process value for the User Archive "workstation". A connection to "Real-time WinCC data" is created in the FoxPro database which is used by a Remote view, "Workstation_count". When the WinCC project is activated, the data is captured and stored in the Remote view, "Workstation_count".

8.2.4 Data transfer from Flat file (.SEQ) and Step 7/WinCC tag

The Inquire module performs the task of monitoring the movement of the pallets across the eight stations, monitoring the status of the latches at the eight stations and providing the information about the next arriving pallet at each of the eight stations.

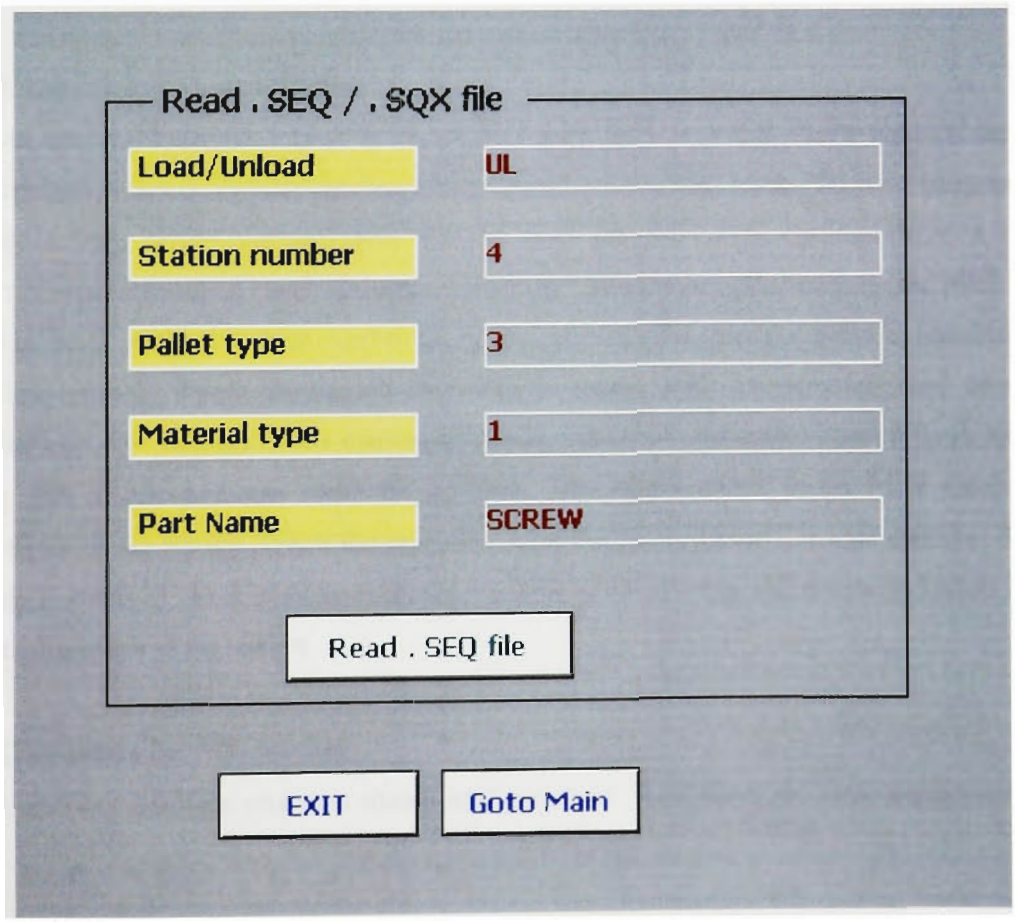


Figure 8.9: Read_SEQ WinCC screen

The Inquire module also has the task of stopping or releasing a pallet at a particular station when requested by the Pallet Controller. The Step 7 "Inquire" program needs to know which pallet is required at a particular station. The ROUTER WinCC program provides this information by reading the ".SEQ" file, thereby demonstrating the intent of this work is to demonstrate the communication between ".SEQ", ".SQX" files and Step 7 tags.

The communication between the "Inquire" program and the ".SEQ" file is achieved by the Global Script of WinCC. (Refer to Appendix D for step-by-step instructions to write data from SEQ and/or .SQX to WinCC tags). A Graphics "Read_SEQ.pdl" (refer to figure 8.9) is created to demonstrate this work. A Project Function "infile()" is created that will open the .SEQ file and assigning it to the file pointer "Pinfile". The infile() function is called up on the mouse-click event of "Read SEQ file". The remaining part of the script scans the .SEQ file and writes the value to WinCC or the Step 7 tag.

8.3 Dataflow from MES to ERP

A Data Transformation Service (DTS) package could be scheduled at the end of each day that would update the ERP database with order status information.

8.4 WinCC Interface for Workstations

This section we will review some interface screens that was designed for CAMCELL

8.4.1 WinCC interface for Lathe Station

The Lathe station operator interface screen is shown in Figure 8.10. It is the screen used by the lathe station operator that helps him or her to monitor the Work in progress and machine status. Work-in-progress information (such as the "part name", "pallet required" and "Current step" information) is obtained directly from the .SEQ file, whereas the "Quantity required" and "Quantity remaining" data can be collected from the MES database. The "Operator message box" generates alarm conditions, if any, to notify the operator about any malfunctioning that takes place at that station. These alarms are user-defined alarms and can be configured according to the requirement of the user. For example, if the coolant level goes below the minimum required level, then an alarm is generated along with a short message about the problem. This data is stored in the MES database for future performance analysis of the station. When the machine status button is "Green", it indicates that the machine is currently running. A status of "Red" indicates that the machine is not running, and the status button flashes yellow when there is a malfunction at the station.

8.4.2 WinCC interface for Mill Station

The Mill station operator interface screen is shown in Figure 8.11. It monitors the same parameters as the Lathe operator screen.

8.4.3 WinCC interface for Vision Station

The Vision station operator interface screen is shown in Figure 8.12. The screen displays Work in Progress and Work Order in queue information. An accept job will be indicated "Green", whereas a reject will indicate "Red" on the screen. The WIP information can be obtained from the same tag that reads the .SEQ or .SQX file, whereas the Work in queue information is obtained from the Router program. The quality of information in the order is updated in the MES database.

8.4.4 WinCC interface for Shop Supervisor

The Shop Supervisor interface screen is shown in Figure 8.13. The screen gives the supervisor an overview of parts being current produced on CAMCELL.

8.4.5 WinCC interface for Shop Manger

The shop manager interface screen is shown in Figure 8.14. The screen gives the manager a bigger picture of the status of CAMCELL. It displays the Scheduler information that includes Machine status and Raw material status. The machine status could be Busy, Idle Or Break Down. The raw material data would come from the MES database.

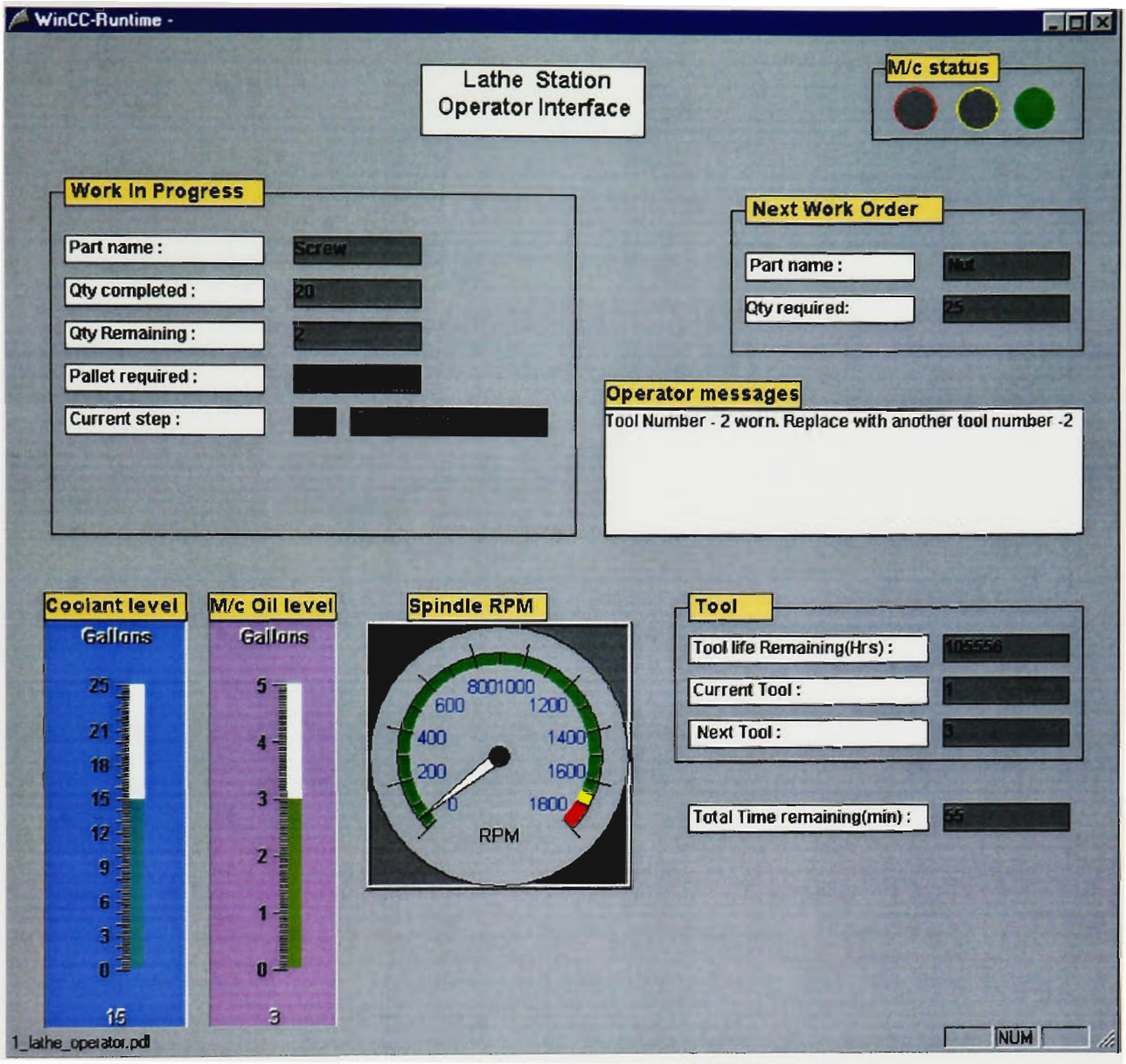


Figure 8.10: Lathe Station Operator Interface

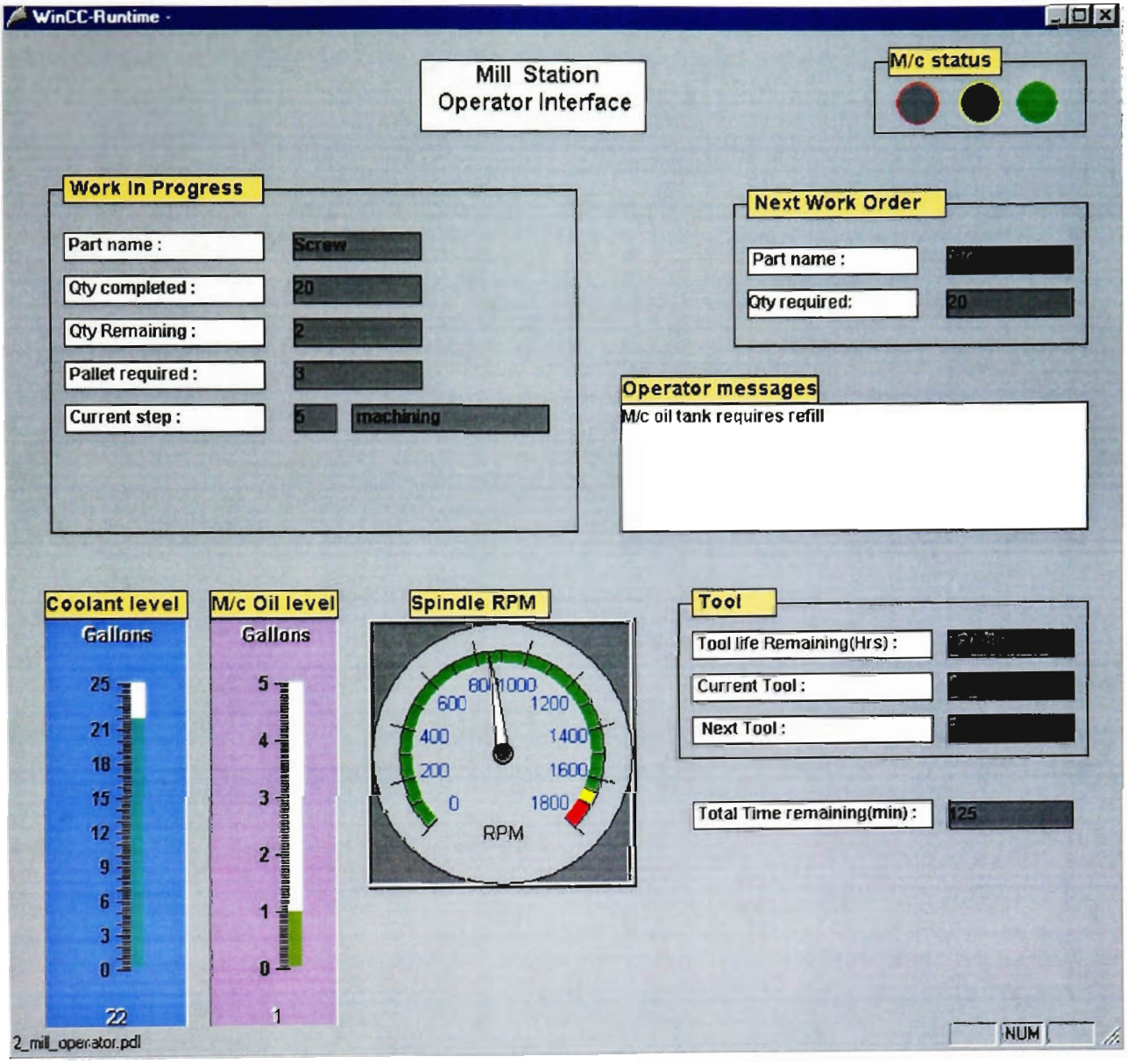


Figure 8.11: Mill Station Operator Interface

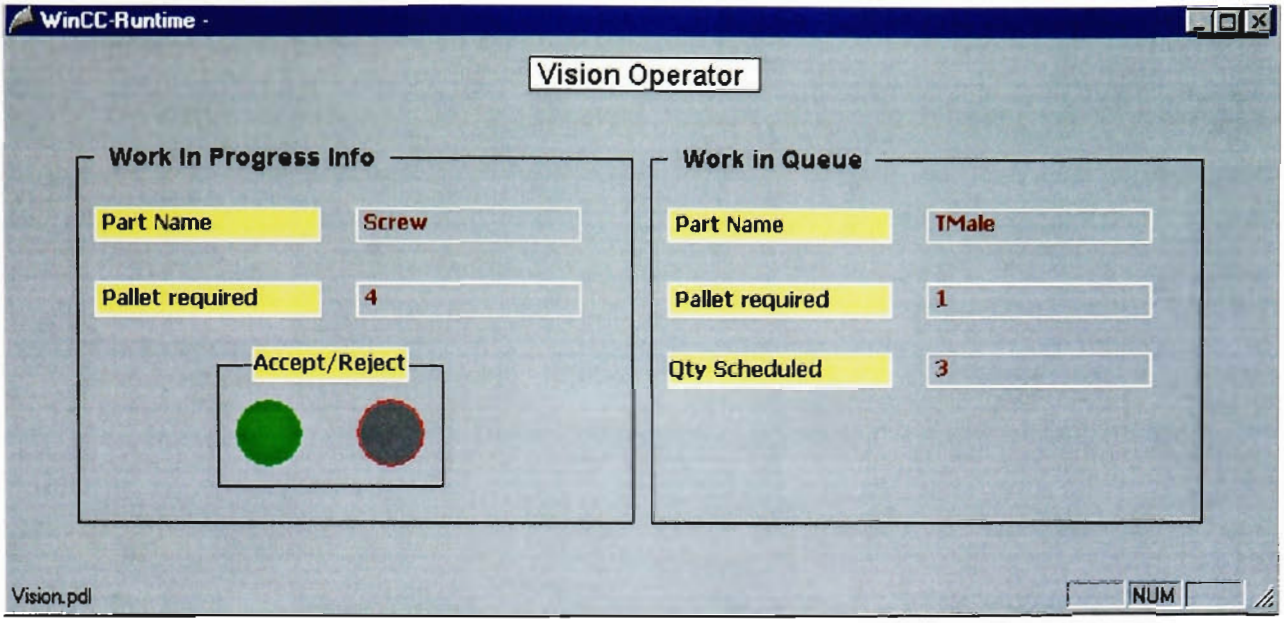


Figure 8.12: Vision Station Operator Interface

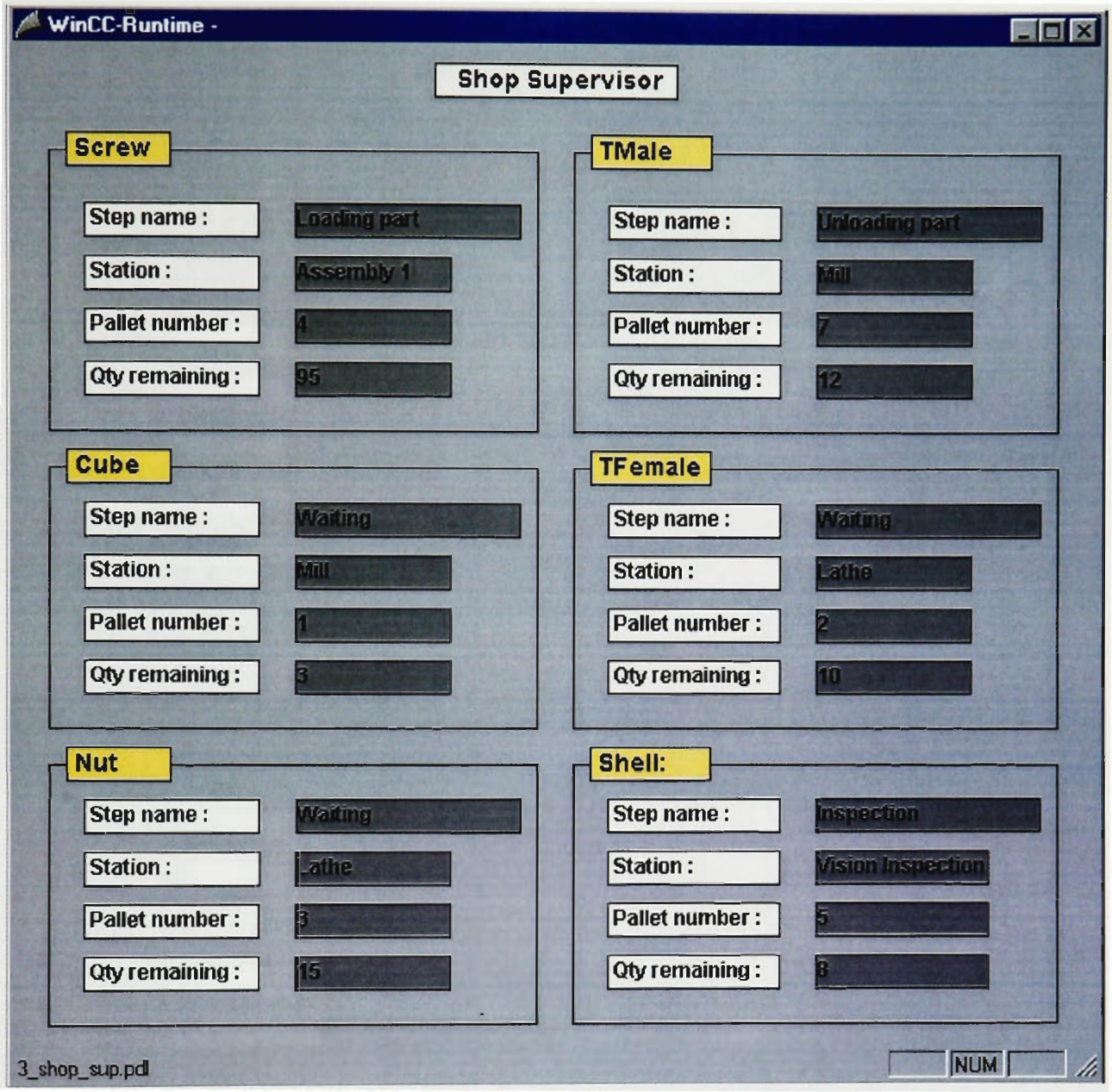


Figure 8.13: Shop Supervisor Interface

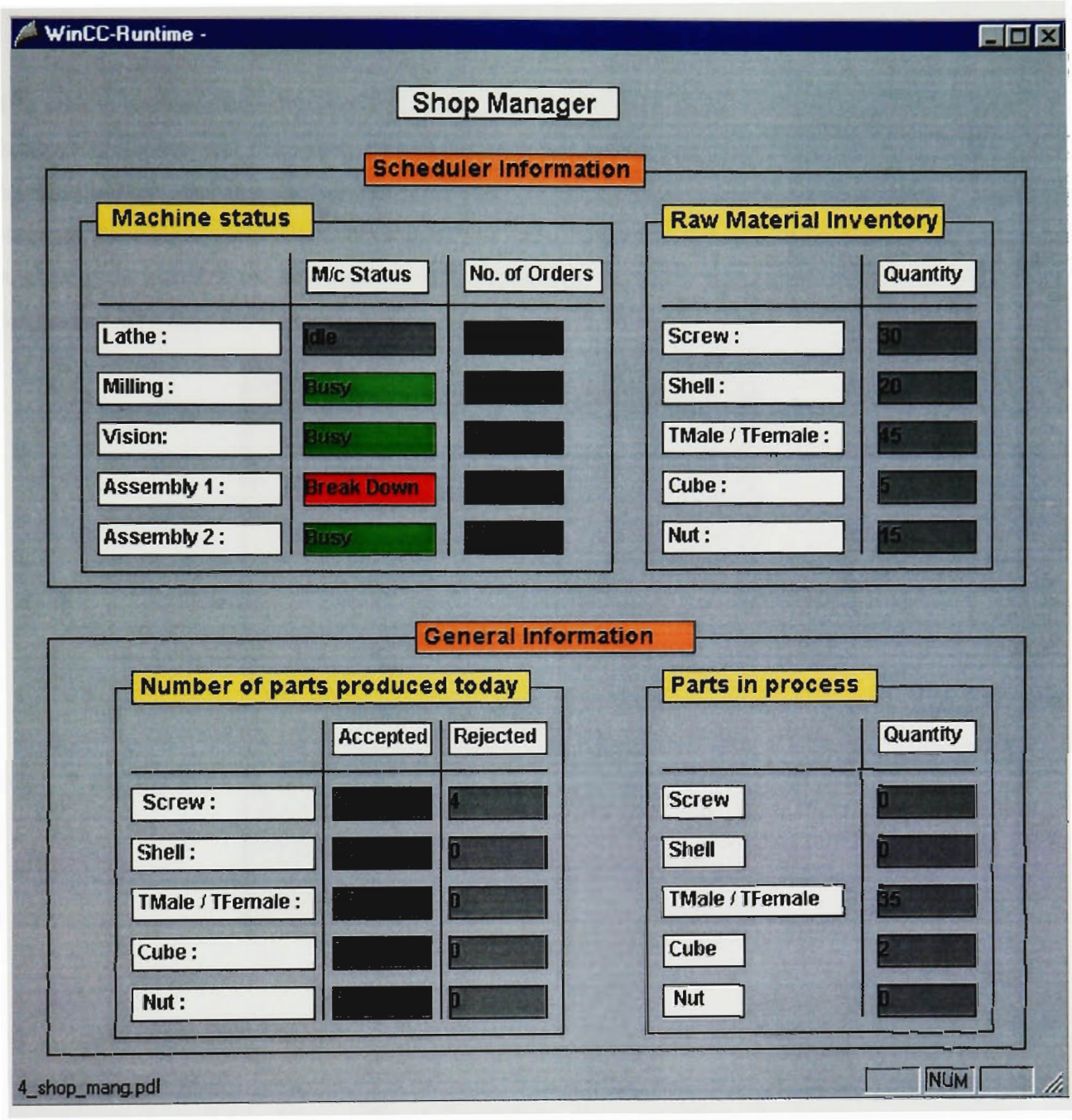


Figure 8.14: Shop Manager Interface

Section 9. VFP screen for Database

This section discusses the proposed VFP Database screens for various users in the system. The MES database is divided into two parts, one to pull real-time data from the control layer of the MES database and the other to push aggregate data to the enterprise level of the MES database. The Shop Manager, Shop Supervisors, Operators and the Customers can access the aggregate data of MES.

A main menu screen is the first screen where appropriate users select the options according to their designation. The main menu screen is shown below in Figure 9.1

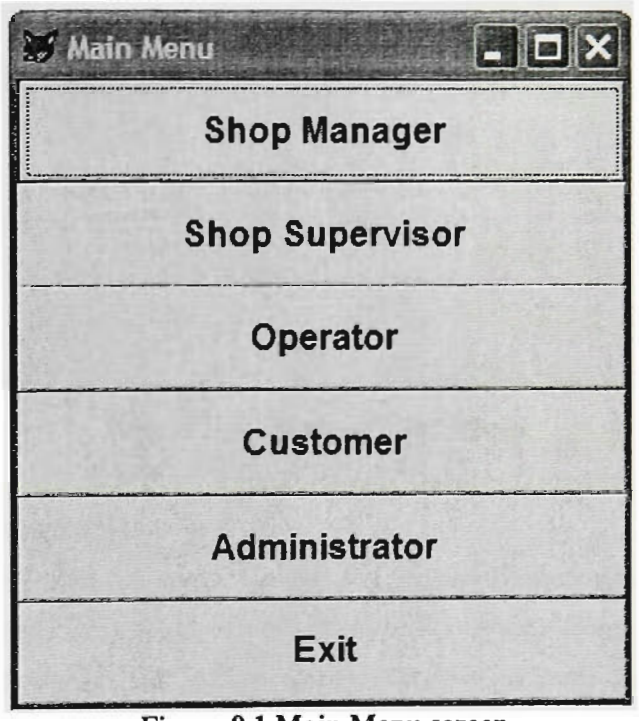


Figure 9.1 Main Menu screen

9.1 Screen for Shop Manger

When the Shop Manager logs onto the system, an authentication screen appears when any user selects the option from the Main Menu (refer to Figure 9.2). The next screen that appears is the "Shop Manager Menu" as shown in Figure 9.3.The Manager can view the following types of data:

- New Work Order
- Schedule Analysis
- Order Status
- Status Analysis
- Daily Summary
- Weekly Summary
- Monthly Summary
- Performance Data
- Resource Status

- Maintenance Scheduled
- History

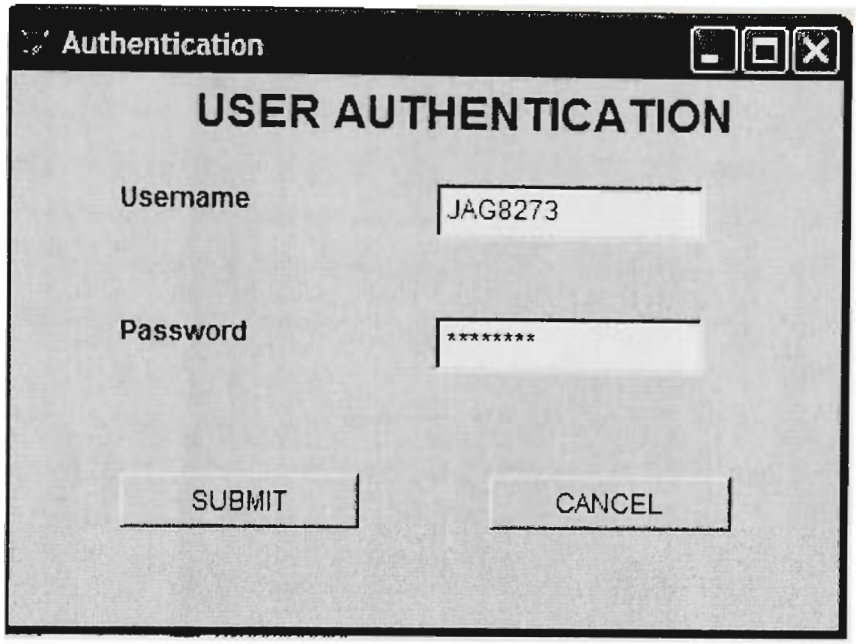


Figure 9.2 Authentication screen

9.1.1 New Work Orders Screen

The New Work Orders screen, as shown in Fig. 9.4, displays information regarding new orders entered in the MES system. The manager can select the Part Name from a list box and select the appropriate date. The "required date" is the customer required delivery date, and the "work order release date" is the date when the work order was released. This information helps the manager to decide on the priority of the orders.

9.1.2 Schedule Analysis

The Schedule Analysis screen (Refer to Fig. 9.5) helps the scheduler or the manager to view the work orders already scheduled for each workstation. The Workstation name is selected from the list box, and the required period is entered in the "From and "To" text box. A graphical view of the work order schedule is shown on the screen. The work order number is shown on the left side of the graph, and the right side of the graph indicates the number of hours for which the order is scheduled on the selected workstation. On clicking any one of the bars in the graph, detailed information--such as Part Name, Part ID, Quantity Scheduled and Scheduled Completion date--are displayed at the bottom of the screen.

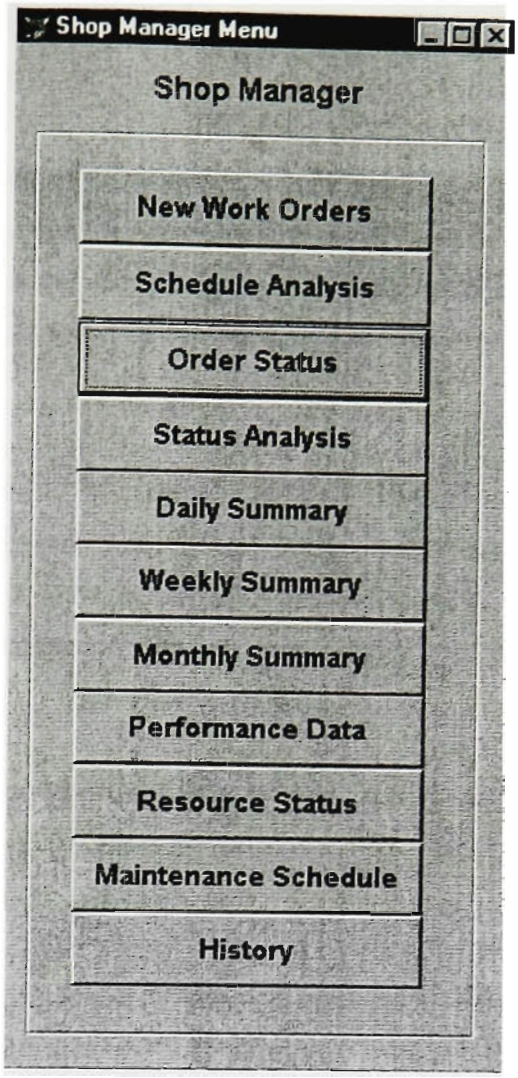


Figure 9.3 Shop Manager Menu

9.1.3 Order Status Screen

The Order status screen is shown in Figure 9.6. This screen displays information for all of the orders in the MES system. When the user clicks on a record, the Work order number and Order date changes in the Text Box, showing the details of that order. The screen displays information, such as Work order number, Order date, Part Name, Part ID, Quantity Scheduled, Qty Completed, Current status of the order, Scheduled Completion date, Actual Completion date and the Due date if the order is still in process. Addition comments about the order can be obtained from bottom of the screen.

9.1.4 Status Analysis

The Status Analysis screen gives the manager an overview of a particular station. He or she selects the desired workstation from the selection box and enters the date/time range. Parameters such as machine set-up time, production time, maintenance time, or machine off time is displayed on the screen. Status analysis screen is shown in Figure 9.7. Clicking any one of the bars in the graph

produces a detailed description of the parameter. The percentage of total hours used is indicated on the right side of the parameters.

9.1.5 Daily/Weekly/Monthly Summary

A Daily, Weekly and Monthly summary report of the production can be obtained that will summarize the total parts produced for that time period. The report will show information such as conformance to schedule, number of defective parts, down time and overall efficiency of the system. Refer to Figure 9.8.

9.1.6 Performance Screens

If the "Performance data" option is selected from the "Shop Manager Main Menu, then the screen shown in Figure 9.9 will appear. Here the user can view data related to

- Conformance to Schedule
- Number of Defects
- Down Time
- Overall Efficiency
- Number of Parts Produced.

Weekly, monthly, or yearly data can be viewed for any of the above requirements in the form of a written report.

A sample of Conformance to the schedule screen is shown in Figure 9.9. The screen gives Graphs and the Overall Conformance Data for the selected period. The chart can be viewed in a Bar, Line or a Pie chart with a two- or three-dimensional view.

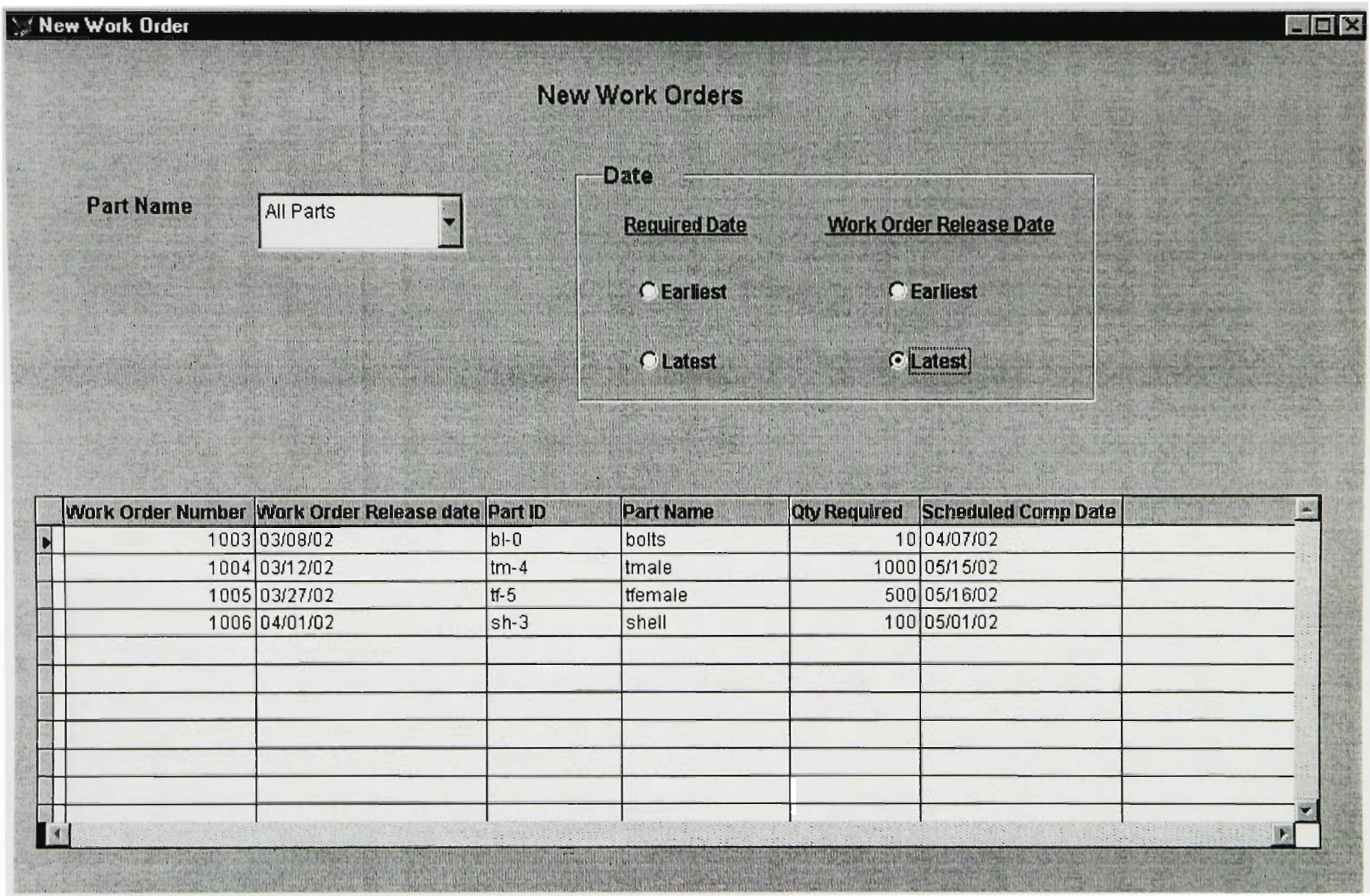


Figure 9.4 New Work Orders (Scheduler/Manager)

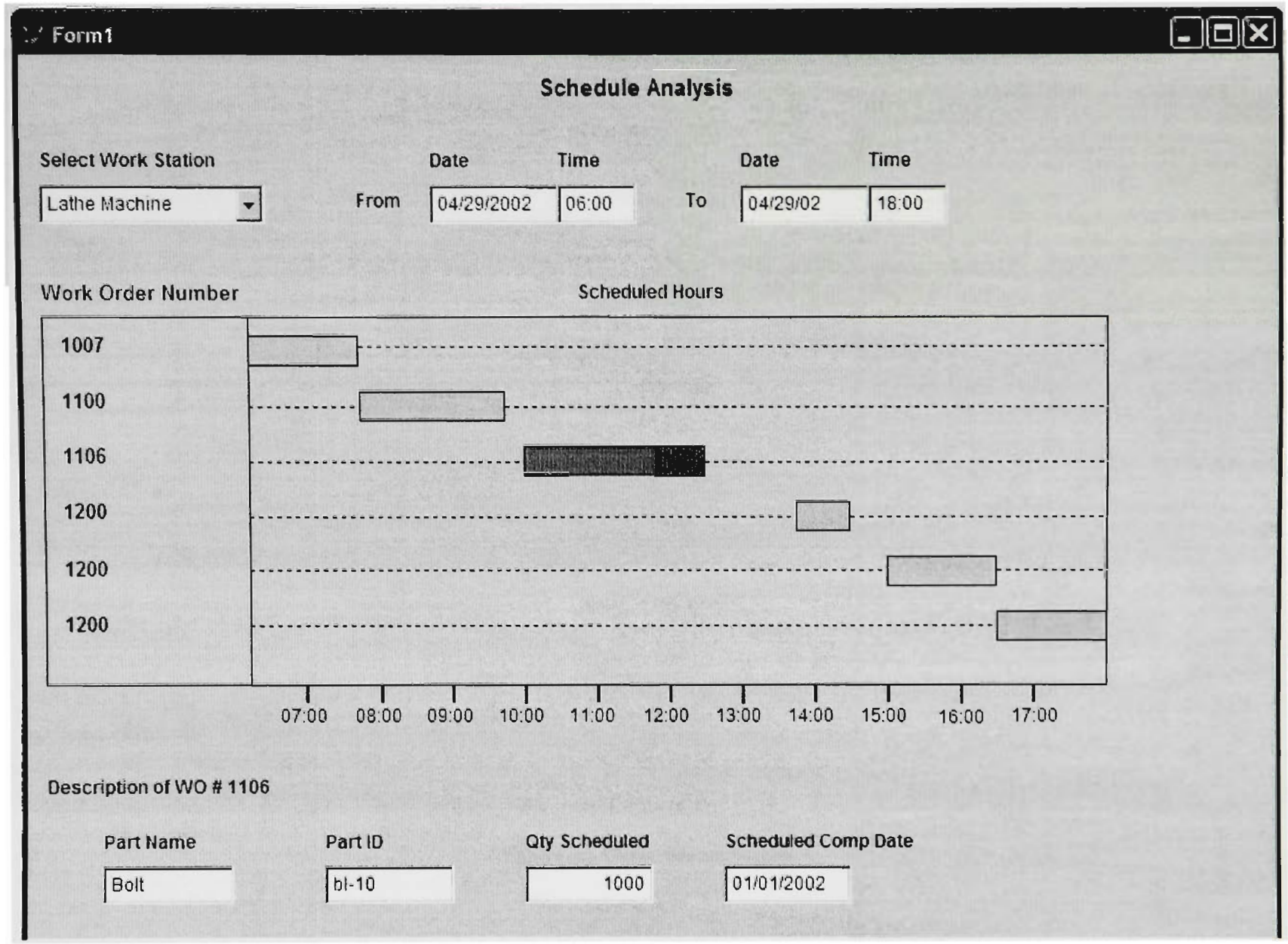


Figure 9.5 Schedule Analysis for Planning (Scheduler/Manager)

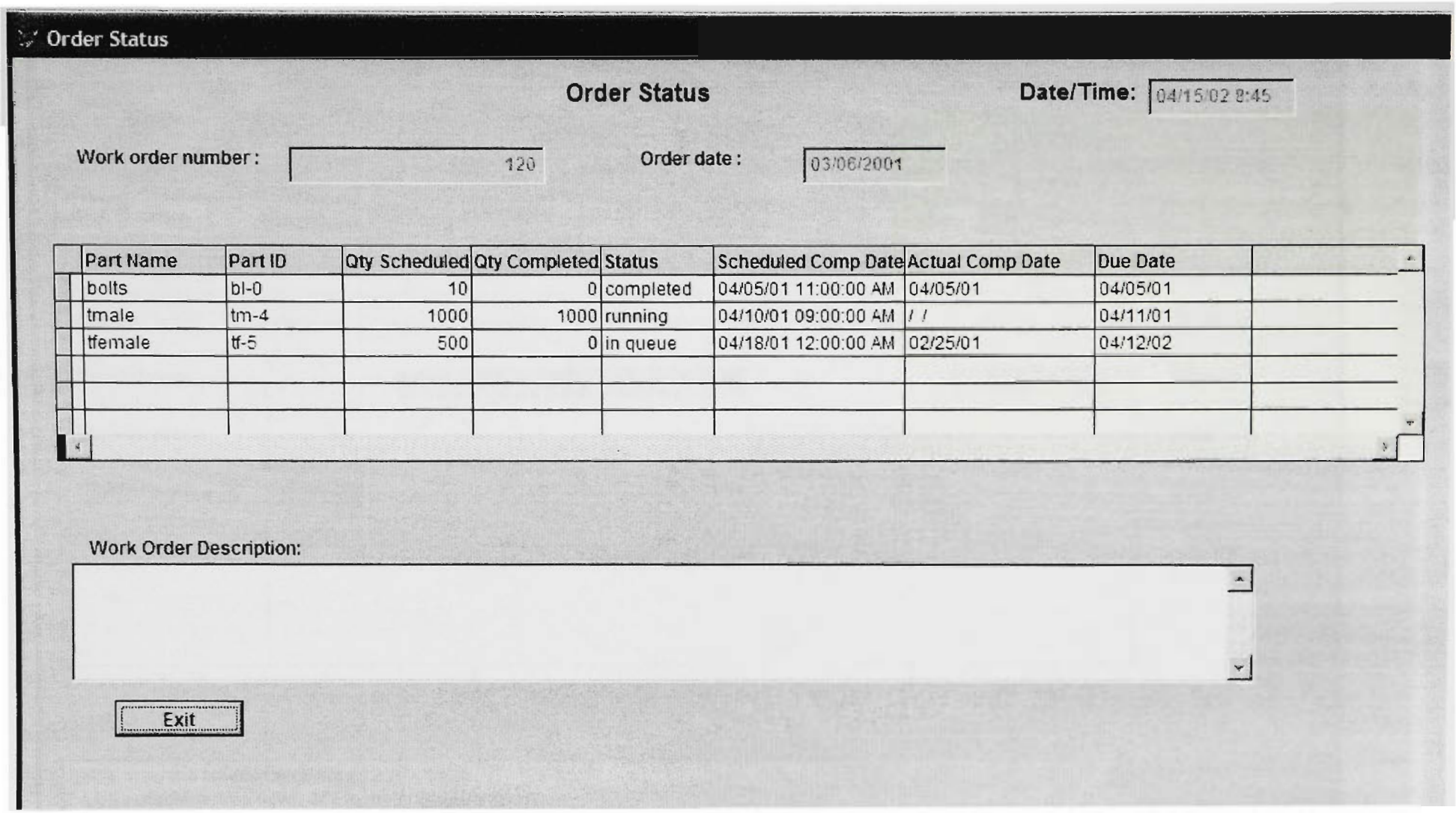


Figure 9.6 Order Status Screen

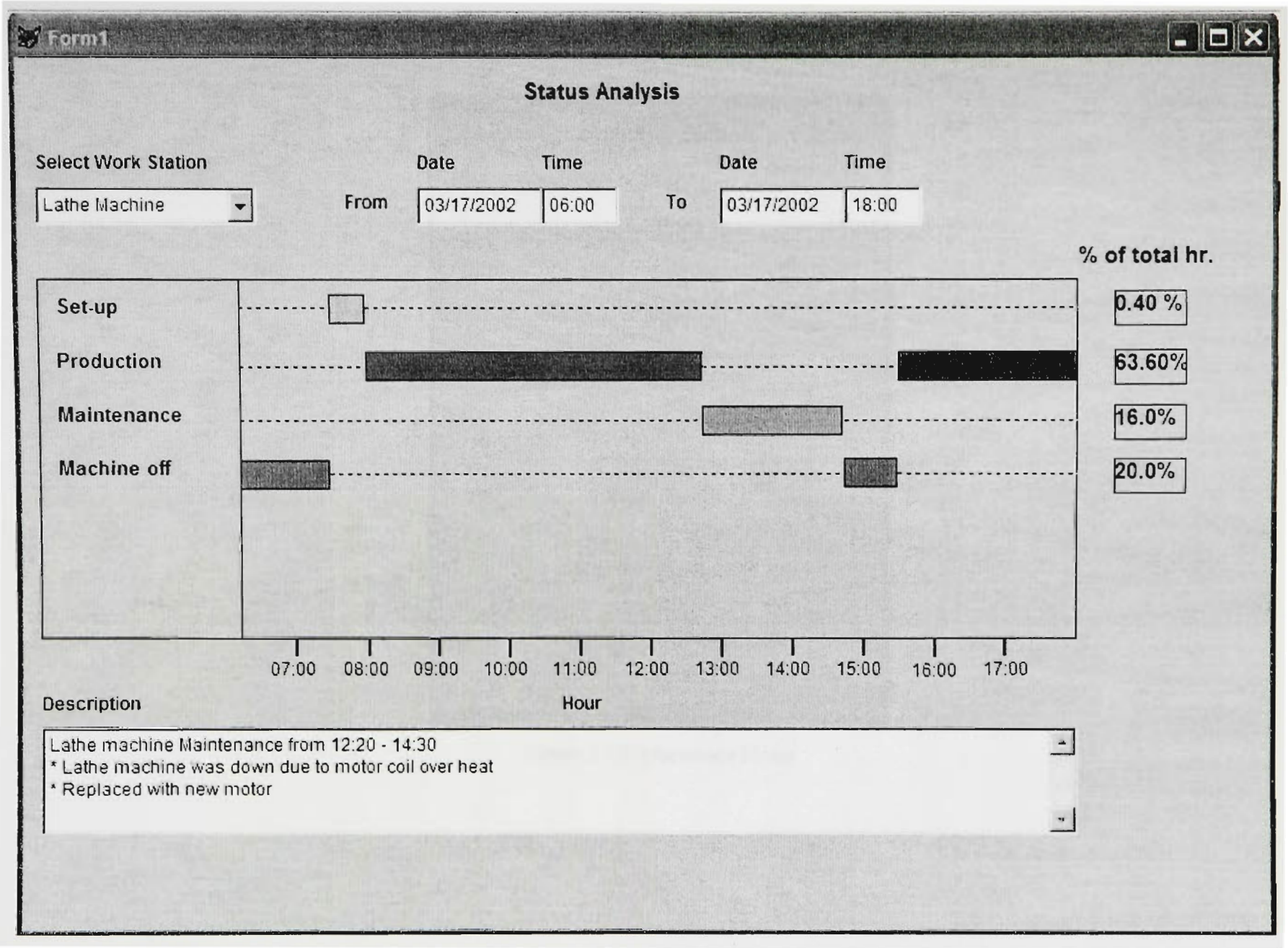


Figure 9.7 Status Analysis

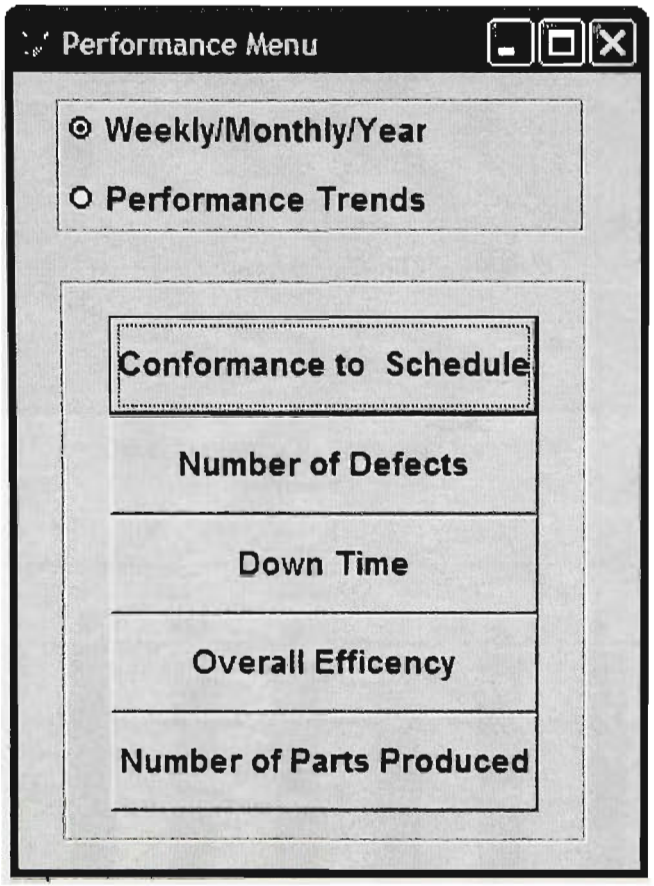


Figure 9.9 Performance Menu

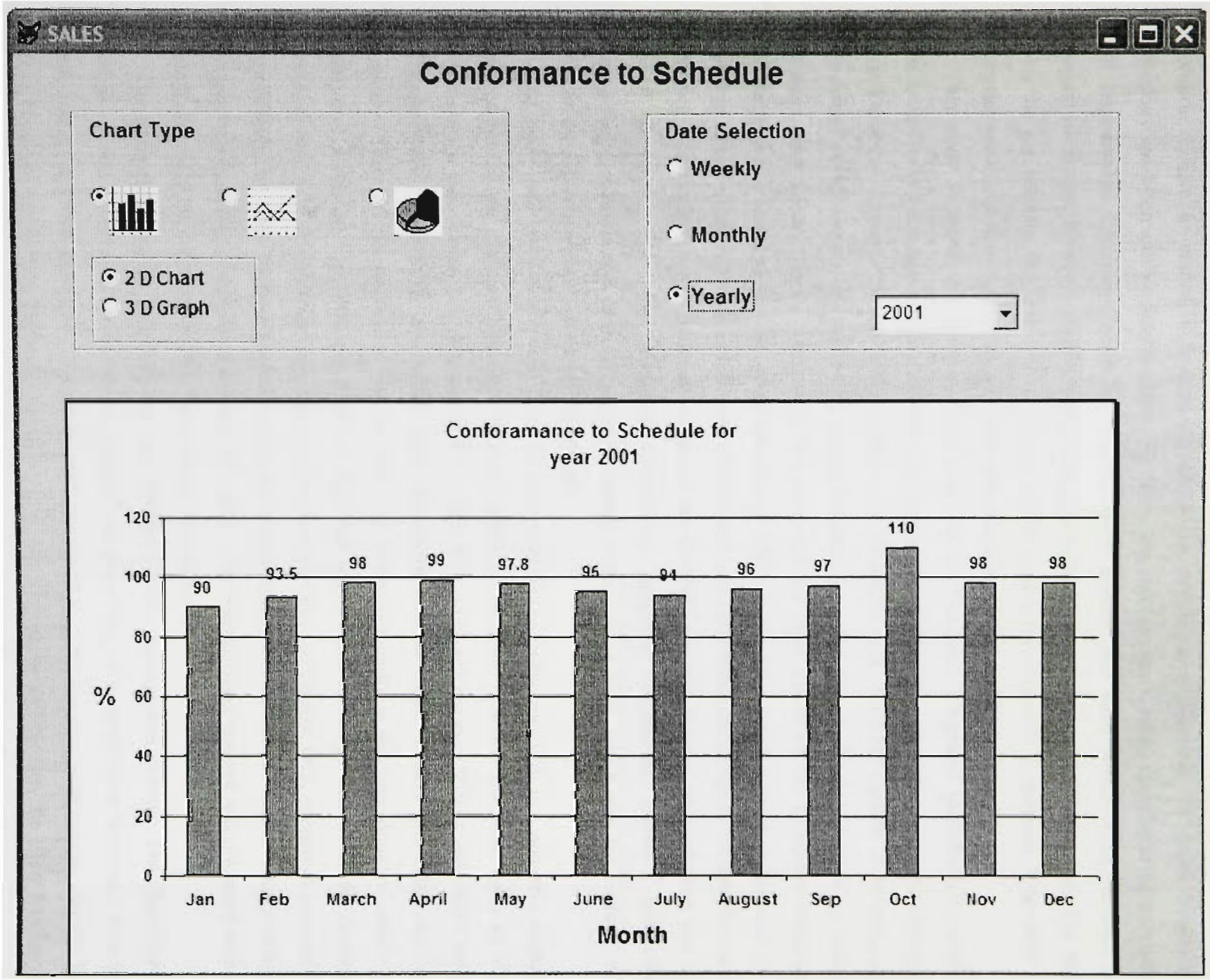


Figure 9.9 Conformance to Schedule

Section 10. Conclusions

The primary focus of this applied research work was to facilitate the development of a Manufacturing Execution System to control a flexible manufacturing system using Siemens' PC-based automation technology and Microsoft's database technology. Over the past few years, many manufacturing companies have implemented Enterprise Resource Planning (ERP) systems and have proved them [meaning the systems] successful. However, retrieving real-time data from the shop floor is a challenging task. The MES closes the gap between the ERP and production worlds so that the execution is optimum. It provides an enterprise-wide, real-time view of the complete manufacturing environment.

This study is aimed at reviewing three layers of the Computer Integrated Manufacturing model: Structured Query Language and Normalization, Client/server methodology for data management systems, Open Database Connectivity and CIM database for Real-time data. An ERP System is an integrated information processing system supporting various business processes, such as finance, distribution, human resources, and manufacturing. This study has reviewed the three layers of the Computer Integrated Manufacturing model and has discussed the role of MES in the information system architecture of a company. It also examined various functions of MES and how these functions complete the data flow in an enterprise's tracking system. The study reviewed the Control layer that focuses on production line and process decisions. The MES transmits instructions, programs, documents, software, and other manufacturing requirements from the support systems to the control layer, which uses the hardware and software, as well as people, to carry out the process.

The study also discussed various types of Structured Query Language. SQL is a powerful database language that is the standard language for many relational database systems. The three types of SQL statements are Data Definition Language (DDL), Data Manipulation Language (DML) and Data Control Language (DCL). The DDL is a set of .SQL commands used to create, modify and delete database structures. They are normally used by the DBA (to a limited extent), a database designer, and/or an application developer. The DML is the area of .SQL that allows a user to change data within the database. It consists of only three command statement groups: Insert, Delete and Update. DCL statements manage the changes made by DML statements. Redundant data wastes disk space and creates maintenance problems. This can be avoided by normalizing the database tables. The normalization concept was discussed with an example showing the tables in first, second and third normal form.

The study discussed the Client/Server methodology for data management systems. The Client/Server systems operate on network environments, splitting the processing of an application between a front-end client and a back-end processor. The study reviewed the three Client/server architectures, namely File server, Database server and Three-tier architecture. In the File server architecture, the client handles the presentation logic, processing logic and much of the storage logic. The file server is a device that manages files operations and is shared by each of the client PCs attached to the LAN. In the Database Server Architecture, the client workstation is responsible for managing the user interface, including presentation logic, data processing logic and business rules logic. Likewise, the database server is responsible for

database storage, access and processing. In the three-tier architecture, a middle tier is added between the users' system interface, client environment and database management server layer. Often the application programs reside on an additional server, which is referred to as an application server. The study also discussed the factors necessary for the successful implementation of client/server projects.

In this thesis work, WinCC applications were developed for CAMCELL, which connects to the FoxPro database using the Remote View and ActiveX Data Objects (ADO). These objects use ODBC and Data Source Name (DSN) to connect to the database. ODBC is an interface that enables applications to access data from a variety of database management systems. A Driver Manager sits between the application and the database-specific drivers. The DSN stores information about how to connect to a particular ODBC connection. The purpose of a Data Source is to gather all of the technical information needed to access the data--the driver name, network address, network software, and so on – into a single place and to make the data access transparent to the user. The steps involved in connecting to a database using ADO were discussed. ADO is a part of the Universal Data Access (UDA) technology that provides access to information across the enterprise. Like its predecessor, Open Database Connectivity (ODBC), UDA provides a common interface for communicating with SQL databases.

During this course of study, WinCC and Visual FoxPro database development environment were found to be an ideal tool for developing an MES to control the CAMCELL. WinCC is a Supervisory Control and Data Acquisition Software HMI (Human Machine Interface). It is a tool for process visualization and information availability. The study elaborates on User Archiving and Global Script editors of WinCC that are helpful for capturing real-time data from the CAMCELL into an MES database. WinCC makes it possible to use computer functions and actions to make processes in a WinCC project dynamic. These functions and actions are written in the ANSI-C language. Actions are started by a trigger or, in other words, are started by an initiating event. Functions do not have a trigger and are used as components of actions as well as in Dynamic Dialogs, Tag Logging and Alarm Logging. For the creation and editing of these functions and actions, WinCC includes the editor Global Script. Data from technical processes can be stored continuously on a server PC via the User Archives of WinCC. Plant Floor Integration is realized with WinCC through its open connectivity architecture. Two databases are required for each WinCC project: a Configuration database (CS database) with the configuration data and a Runtime database (RT database), which contains the process data.

The study discussed the Software architecture and the Information flow in the CAMCELL using data flow diagrams. The five-control software modules (Order Manipulation, Scheduling, Routing, Station controller, and Support) were discussed. This thesis work proved the ability of WinCC to capture real-time data from the control layer into the CELL MES database (Visual FoxPro). The study successfully established link to read- and write-to and -from WinCC and VFP databases. Various WinCC screens were proposed for controlling and monitoring the CAMCELL. Similarly, various database screens were proposed for various levels in the organization to retrieve aggregate data from the system.

Overall, the technology was found to be an optimal application development environment for building an MES and for controlling a manufacturing system, and it was true to its claim of having a totally "open" architecture.

Appendix A

Read Data from FoxPro Database and Write to WinCC Tags

Appendix A. Read Data from FoxPro Database and Write to WinCC Tags

Purpose
The purpose of this document is to give step-by-step instructions to read data from FoxPro database and write it to WinCC tags. A graphics screen will show the Order information.

Pre-requisite
Create WinCC tags as per Attachment 1 of this Appendix.

Procedure:
The procedure for collecting real-time data from the WinCC can split into 3 main steps:

I. Creating WinCC graphic for viewing the Database records.
II. Creating FoxPro Database tables and ODBC connection.
III. Writing scripts to read necessary information and transfer them to WinCC tags.

1. Create a main screen, as shown in Fig A.1 that will help us browser through other Operator interface screens. To do this, open a new graphic from the WinCC Control Center Graphics Designer editor. Create the screen as shown below. We will configure the buttons to take us to the respective screens once we have created them. Save this screen as "Main.pdl".

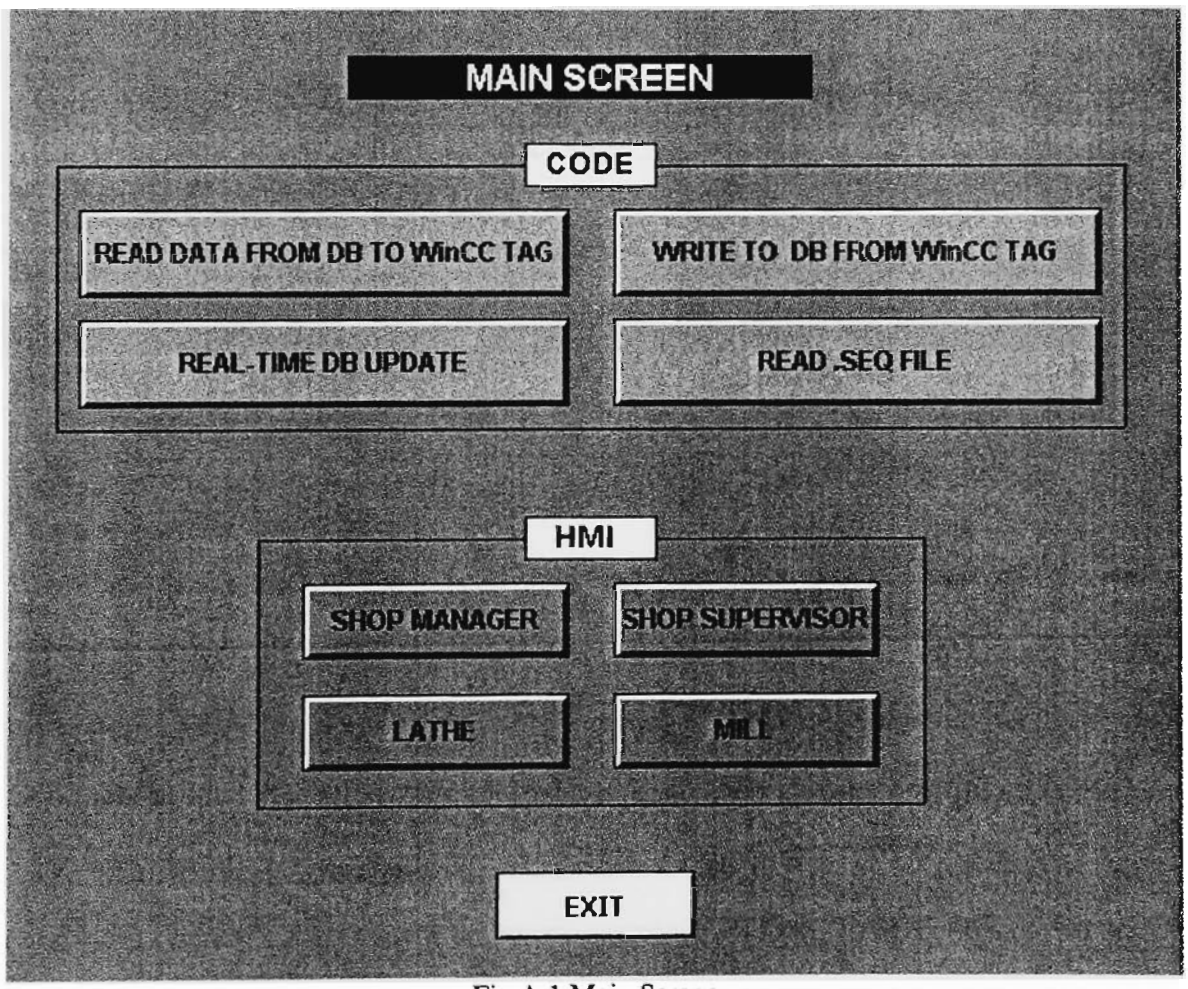


Fig A.1 Main Screen

2. Open a new graphic and create a graphic as shown in Fig A.2. Save the graphic as "Order_data.pdl"

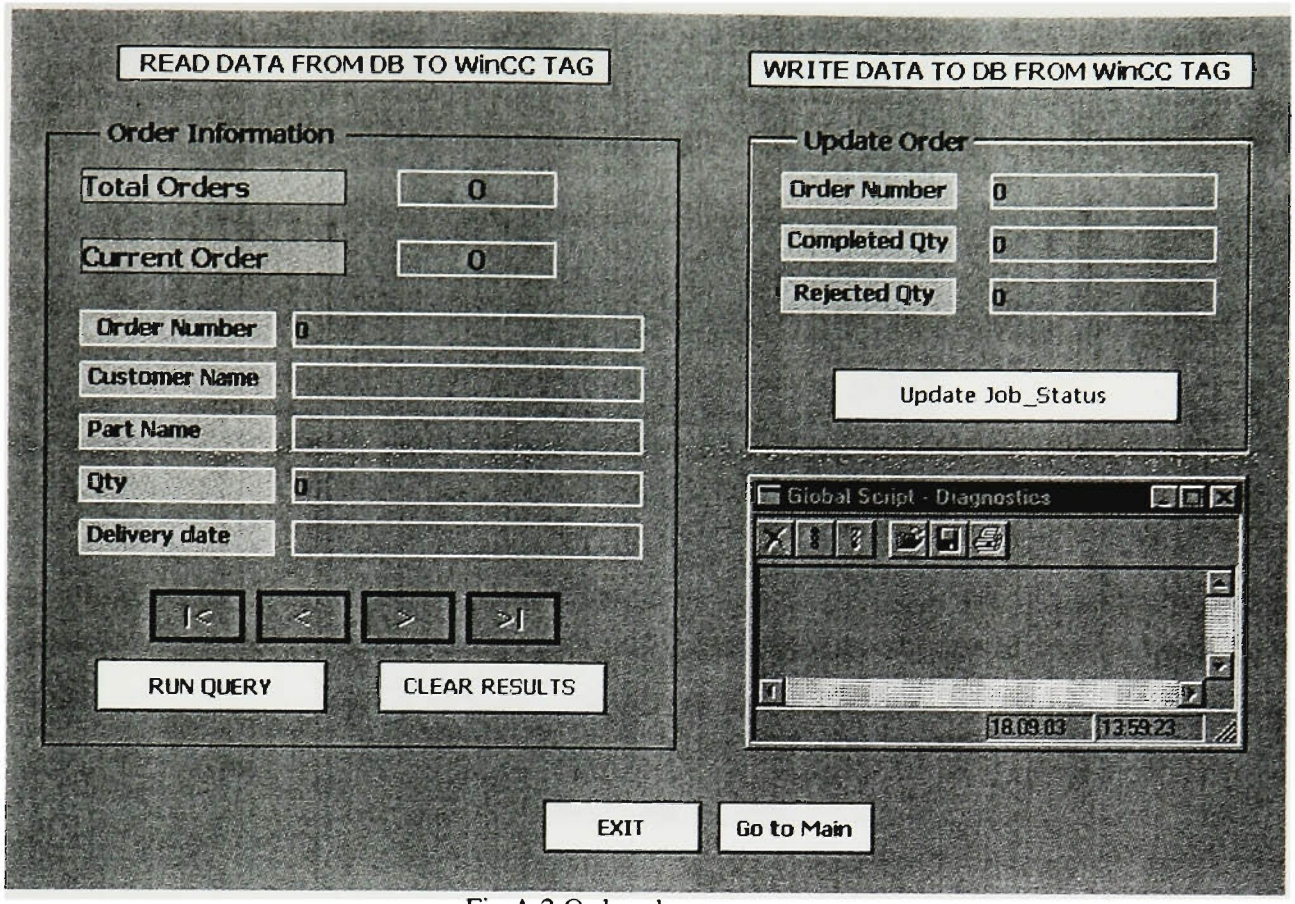
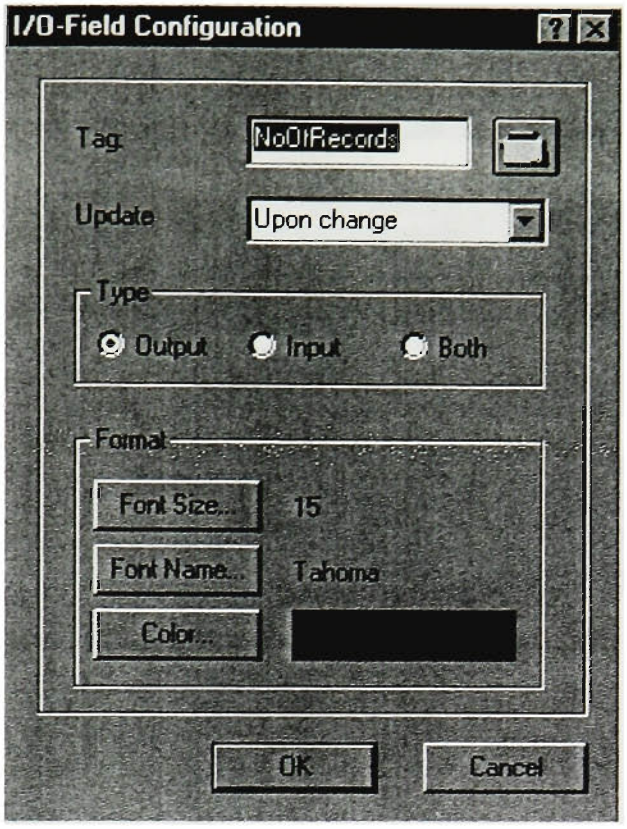


Fig A.2 Order_data screen

Configure the IOField as per the list below:

IOField	Tag	Update	Type
Order Information			
Total Order	NoOfRecords	Upon change	Output
Current Order	CurrentRec	Upon change	Output
Order Number	Order_Number	Upon change	Output
Customer Number	cust_name	Upon change	Output
Part Name	Part_Name	Upon change	Output
Qty	qty_req	Upon change	Output
Delivery date	Req_Date	Upon change	Output
Update Order			
Order Number	Order_Number_UPDATE	Upon change	Input
Complete Qty	Compl_Qty_Update	Upon change	Input
Rejected Qty	Reject_Qty_Update	Upon change	Input

In order to configure the IOField, right click on the IOField and select the option "Configuration Dialog" from the pop up menu.



3. In the properties of the "Run Query" button of the "Order_data.pdl" screen, right click on the "Operator Control Enable" properties in Dynamic column and select Dynamic Dialog option form the pop up menu. Configure the parameters as shown in the Fig A.3 below. By doing this we enable navigation and update buttons only if Run Query button is clicked and disable clear results button.

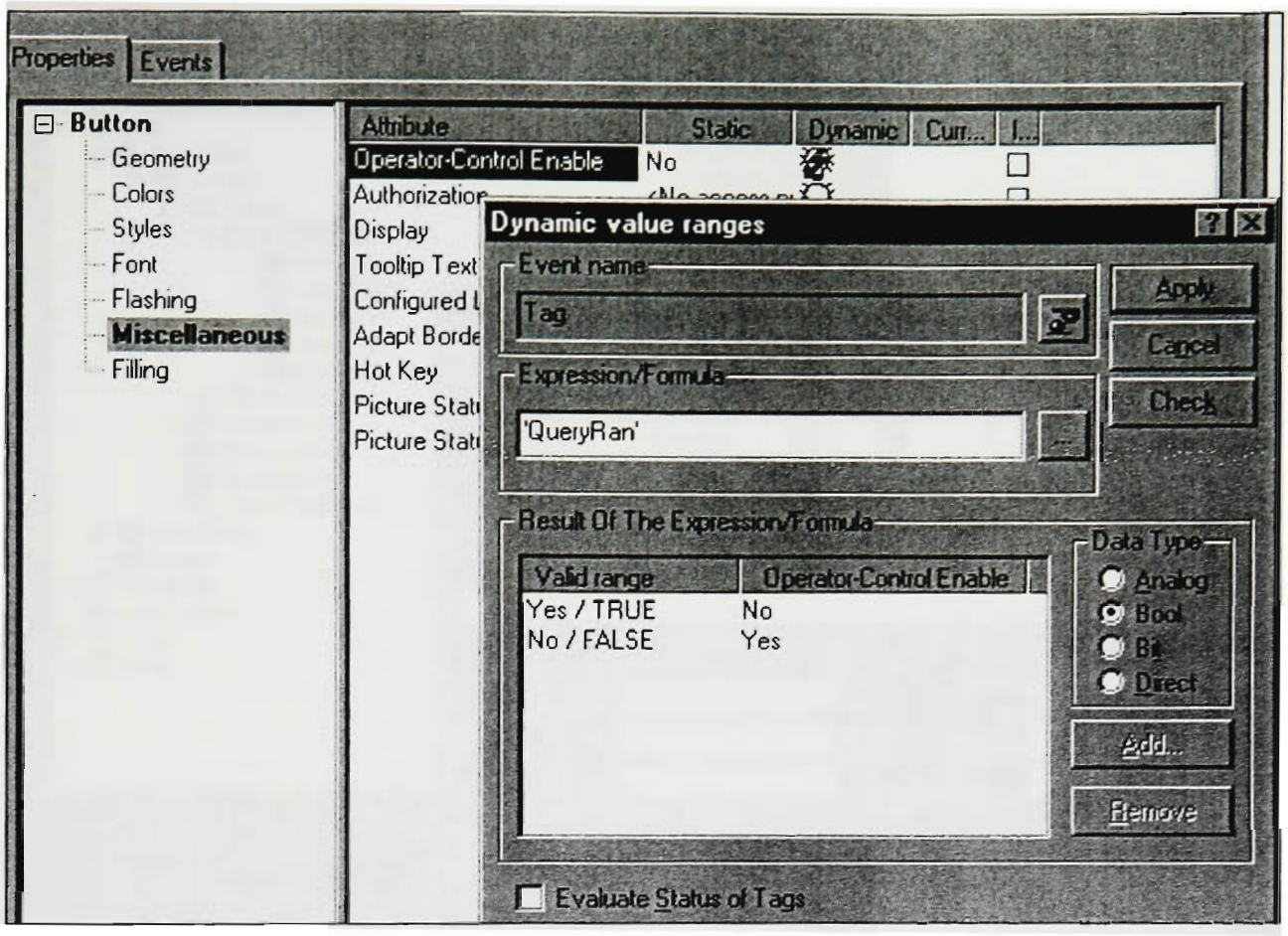


Fig A.3 Properties of Run Query button

4. We will now create the "Order" database that will have "job_status" and "order_data" tables. The "job_staus" table should have field as shown in the figure Fig A.4 below.

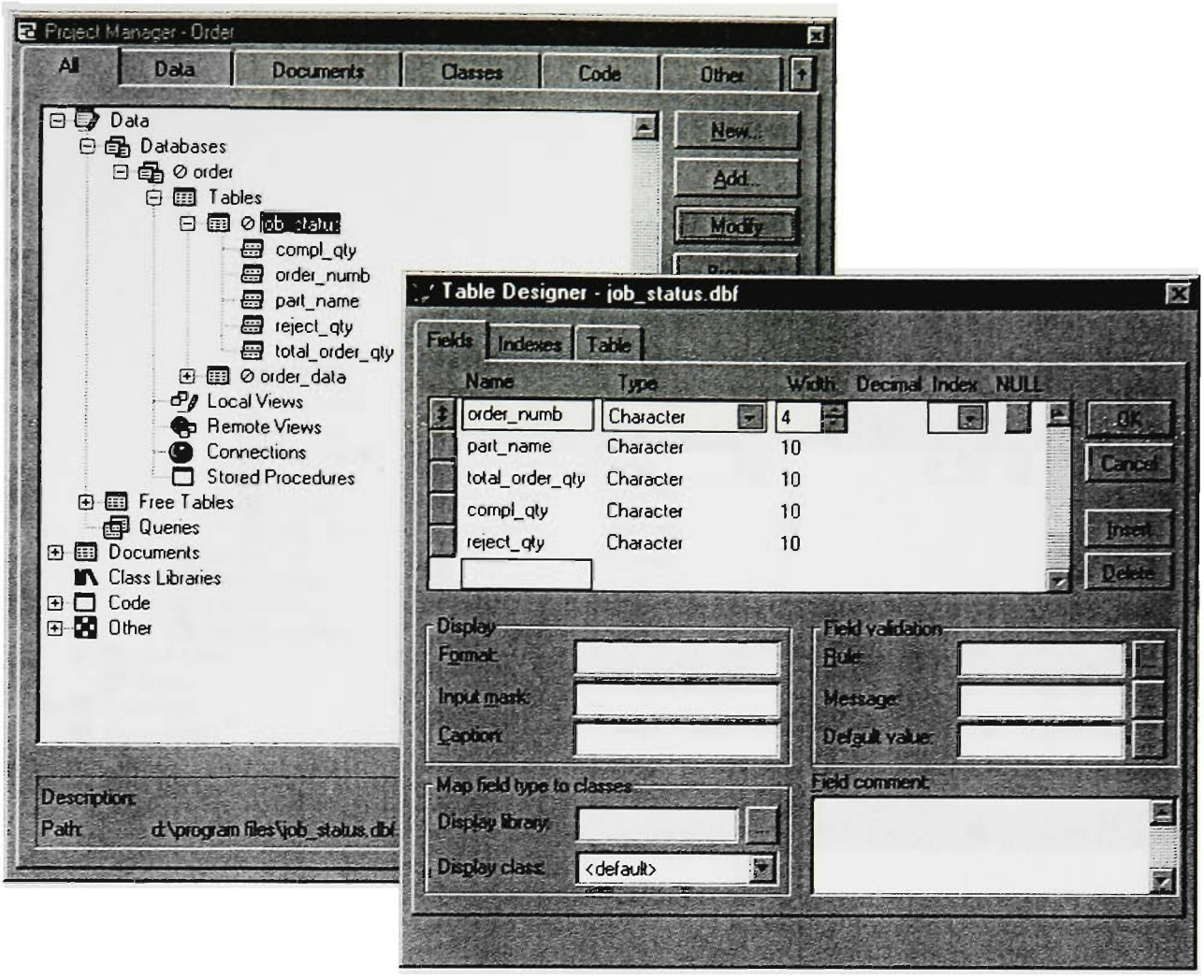


Fig A.4 Job status table

5. Enter sample data into "job_status" table. I have following data entered.

order_numb	part_name	total_order_qty	compl_qty	reject_qty
222	Shell	124	45	1
225	TMale	111	12	0
220	Bolt	200	123	2

6. The "order_data" table should have fields as shown in the Fig A.5 below. We have created the tables necessary for this tutorial.

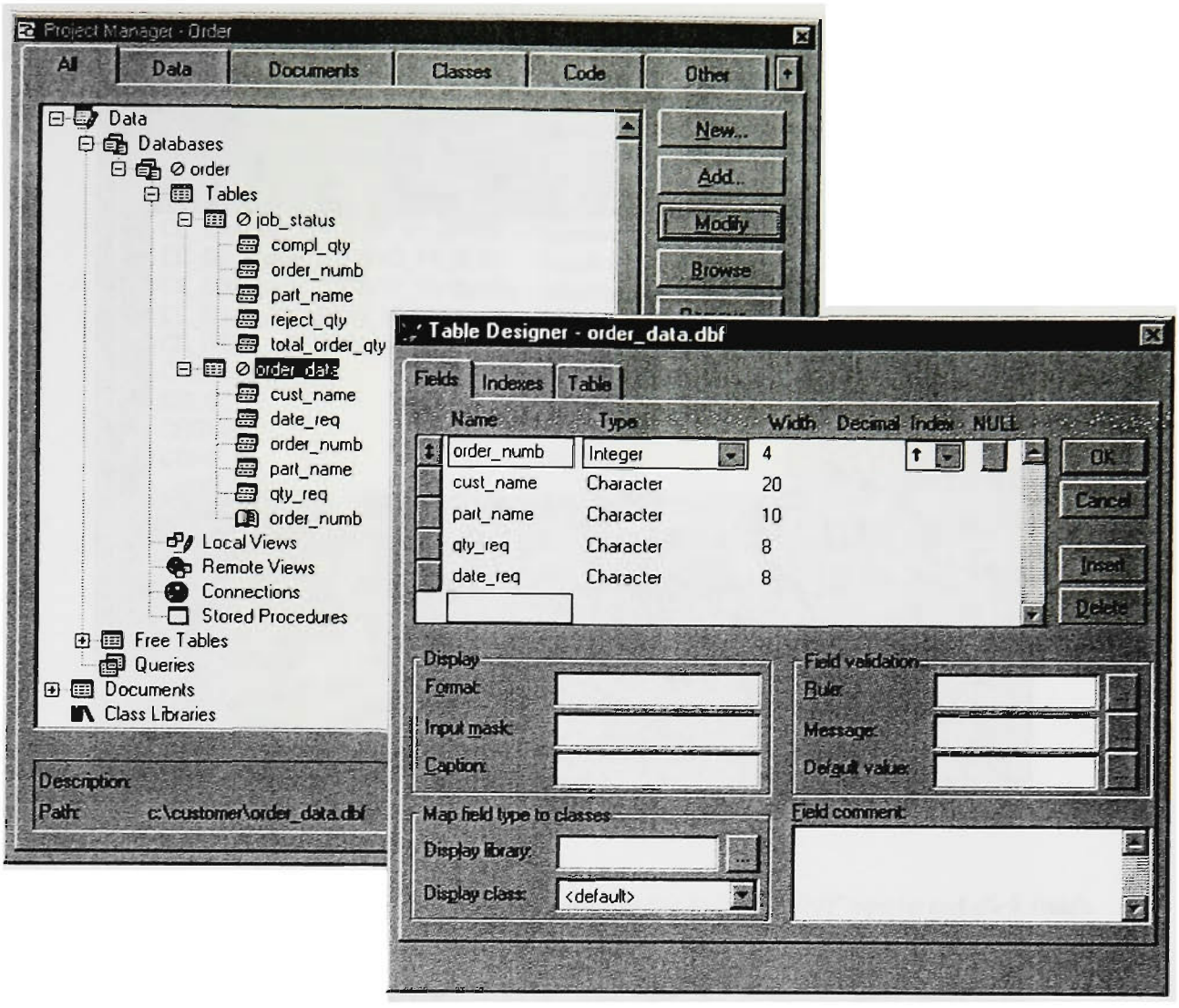


Fig. A.5 Order data table

7. Enter sample data into "order_data" table.

order_numb	cust_name	part_name	qty_req	date_req
222	Kodak	Shell	124	06/08/04
225	Xerox	TMale	111	04/30/04
223	Brook-Anco Corp	Screw	124	05/06/04
221	CDS Manufacturing	Shell	222	06/08/04
224	VR Industries, Inc.	Nut	100	12/12/04
220	SA Inc.	Bolt	200	01/08/04
226	RC Industries	Nut	150	02/05/04

8. In order to establish connection between FoxPro and WinCC we will create an ODBC Data Source Name (DSN). Open the "Data Sources" from control panel. A window similar to one shown in Fig. A.6 below will open.

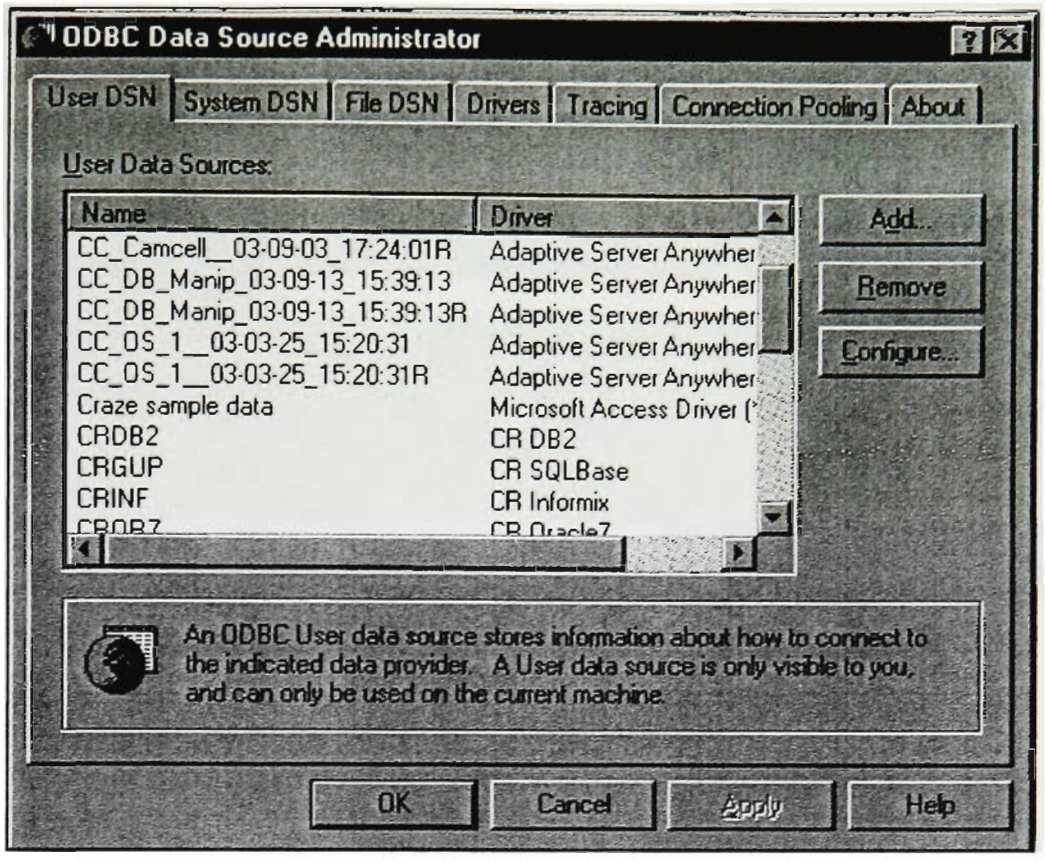


Fig A.6 ODBC Data Source administrator

9. Click Add button and select the "Microsoft Visual FoxPro Driver (*dbf)" option and click finish.

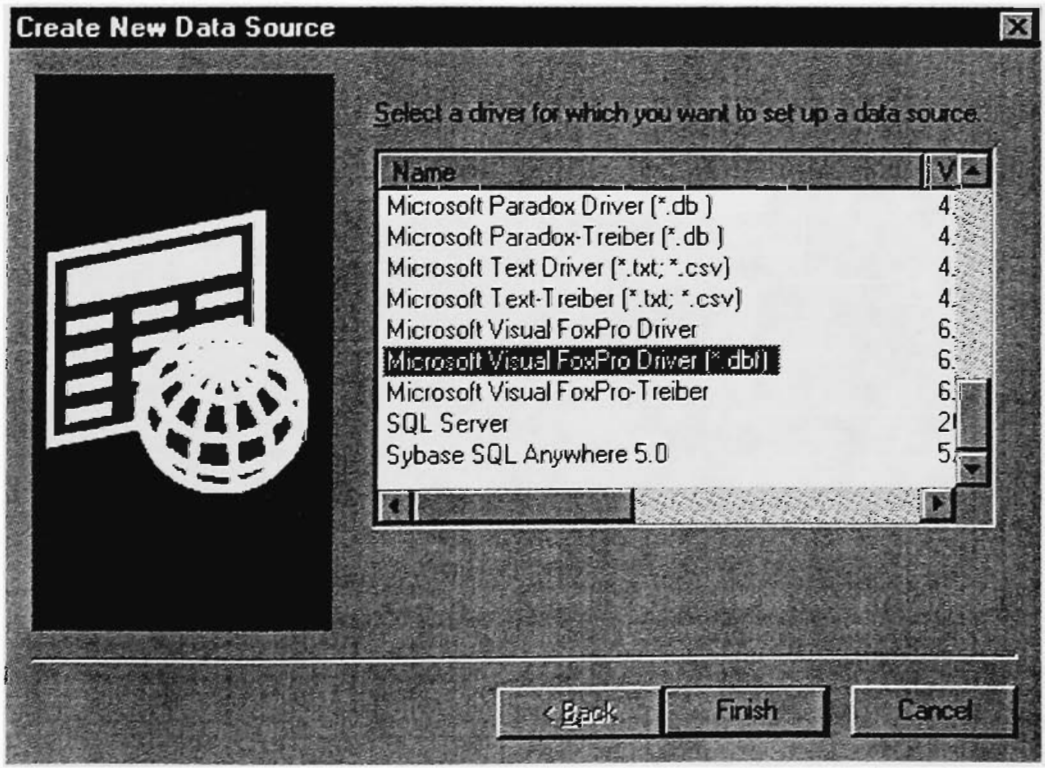


Fig A.7 New Data source window

10. Name the Data Source Name (DSN) as "set" and in the path field of the window, browse to the location of "order" database that was created earlier. Click OK.

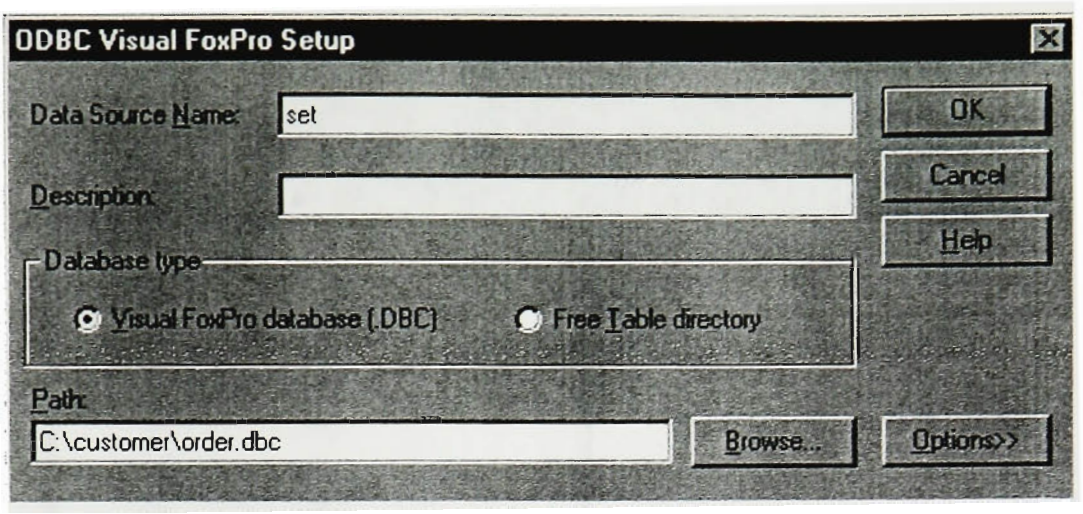


Fig A.8 ODBC Visual FoxPro Setup window

11. The "set" DSN is now created and is seen in the Fig A.9 below.

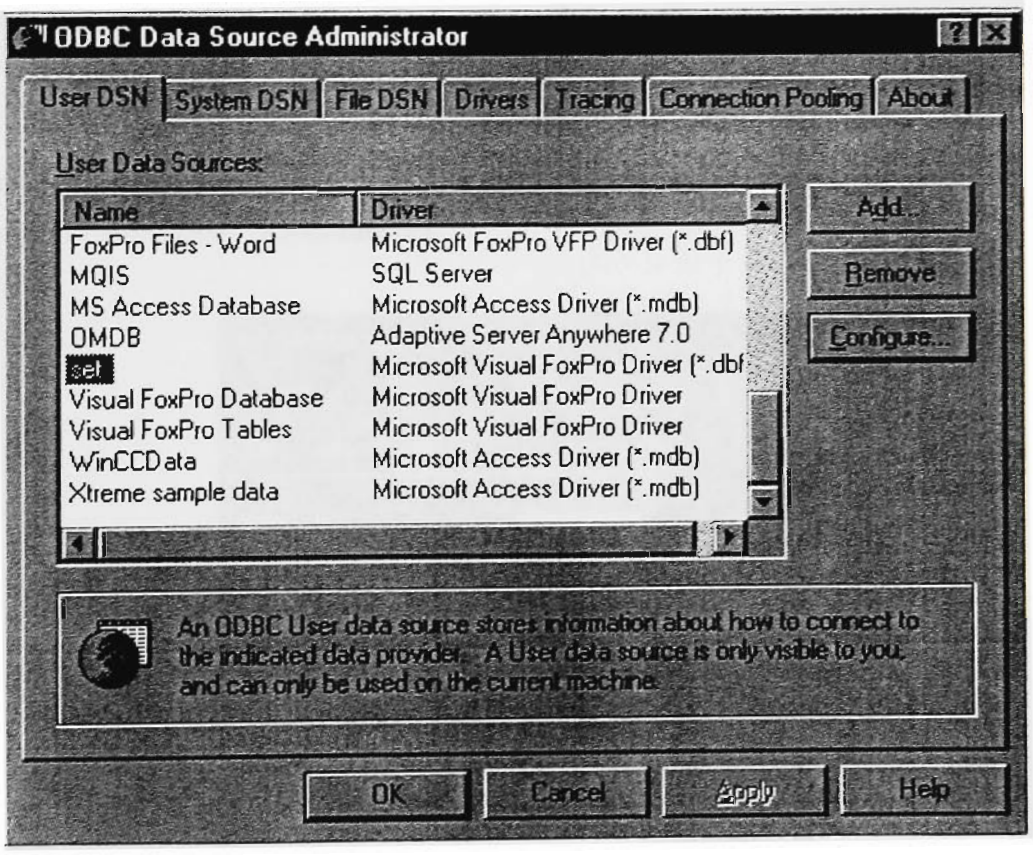


Fig A.9 Window showing the "set" DSN created

12. After we are done creating the database and the DSN, next we write a script that will connect to the FoxPro database via "set" DSN; read the "Order_data" table and feed the data to respective WinCC tag that was created in step 2. A modular approach is taken here by creating a global function, ConnectToDB(), that will open the database using "set" DSN. The other part of the script is written on the "mouse click event" of "Run Query" button. The function ConnectToDB() can be can be called by any other function that needs access to "Order" database.

From the WinCC Control Center open the Global Script editor. To do so right click on the "Global Script" icon and select the Open option form the pop up menu.

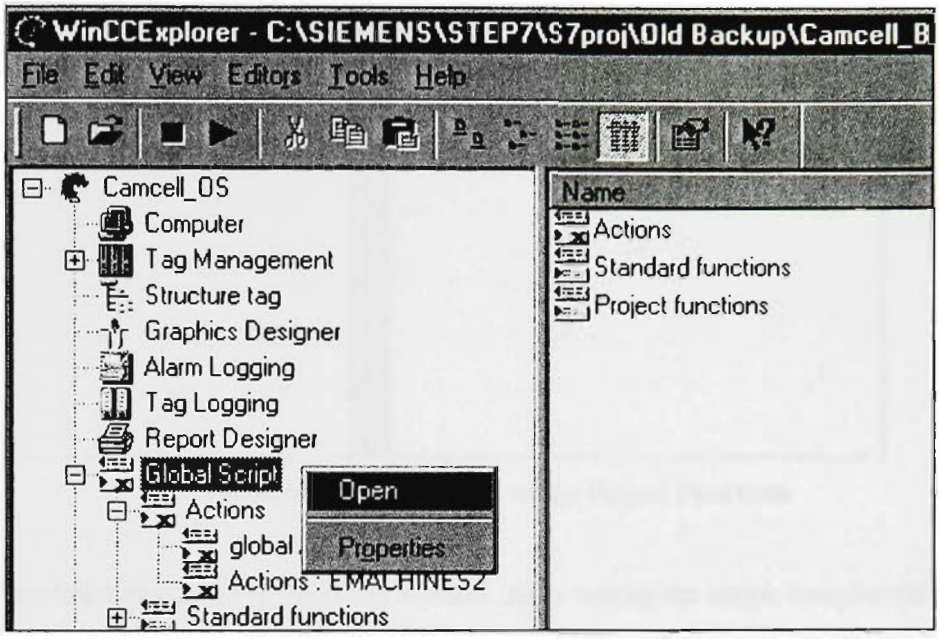


Fig A.10 Global script option in WinCC Explorer

13. Right click on the Select "Project functions" option and select "New" option from the pop up options to create a new "Project function"

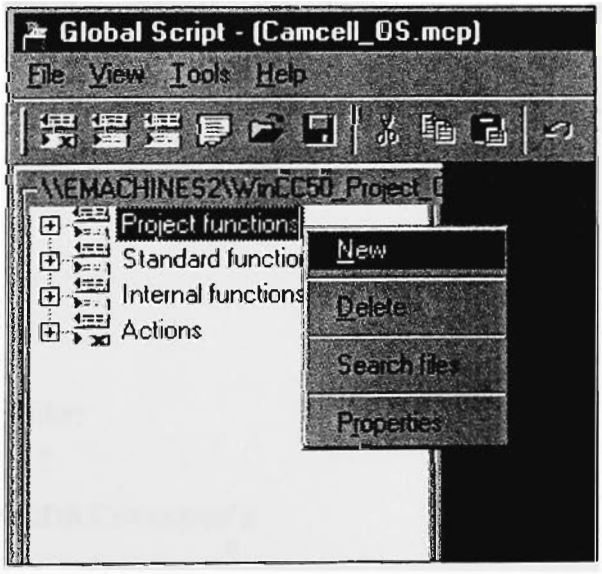


Fig A.11 Global script option in WinCC Explorer

14. A new function editor window will open as shown in Fig A.12 below.

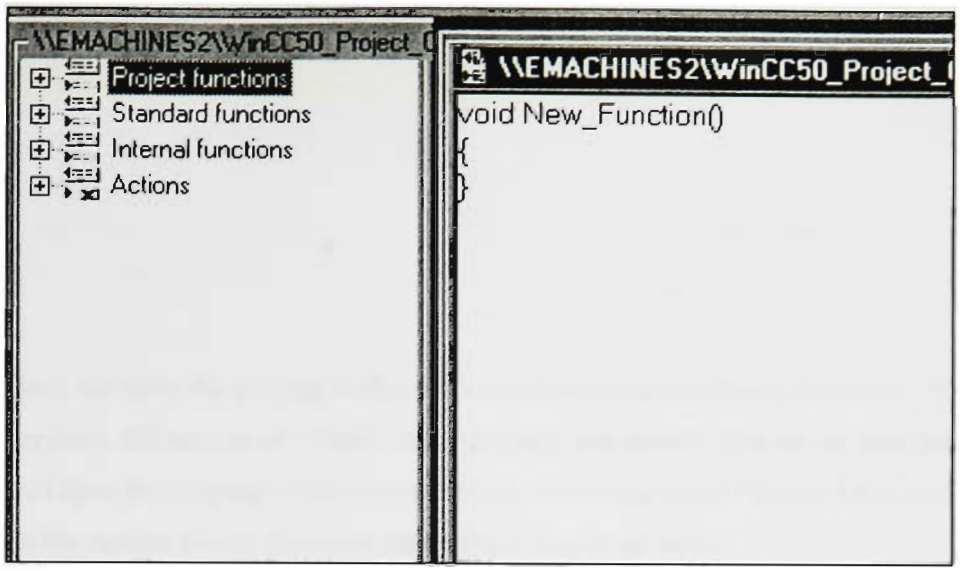


Fig A.12 Function editor within Project Functions

15. Type the following script in the editor window. After writing the script, compile the function by clicking the compile button  on the toolbar and save it as "ConnectToDB.fct"

```
//*************************************************************************************
// This function connects to the DSN "set"
//  It can be called by any script that needs access to "order.dbc" table of "customer" FoxPro Project
//*************************************************************************************

__object *gcn, *grsOrder,*grsOrderUpdate; // these are global  variables that  can be use in any
                                          //function of this project

void ConnectToDB()
{

extern __object *gcn, *grsOrder;

gcn = __object_create("ADODB.Connection");

if(NULL==gcn)
    printf("Create Connection Failed\r\n");
else
    {
    // sets the connection object to use the DSN "Set"  This DSN is already created
    gcn->Open("DSN=set");

    // checks to make sure the connection was successful and that the database was found
```

```
if (gcn->State==0)
        printf("Connection Failed\r\n");

}

}
```

16. Next, we write the C script in the mouse click event of Run Query Button of "Order_data" graphics. Go back to of "Order_data" graphics and double click on the Run Query button. This will open the property window of the button. In Events select "Mouse Press left" and right click on the Action. Select C-Action option from the pop up menu.

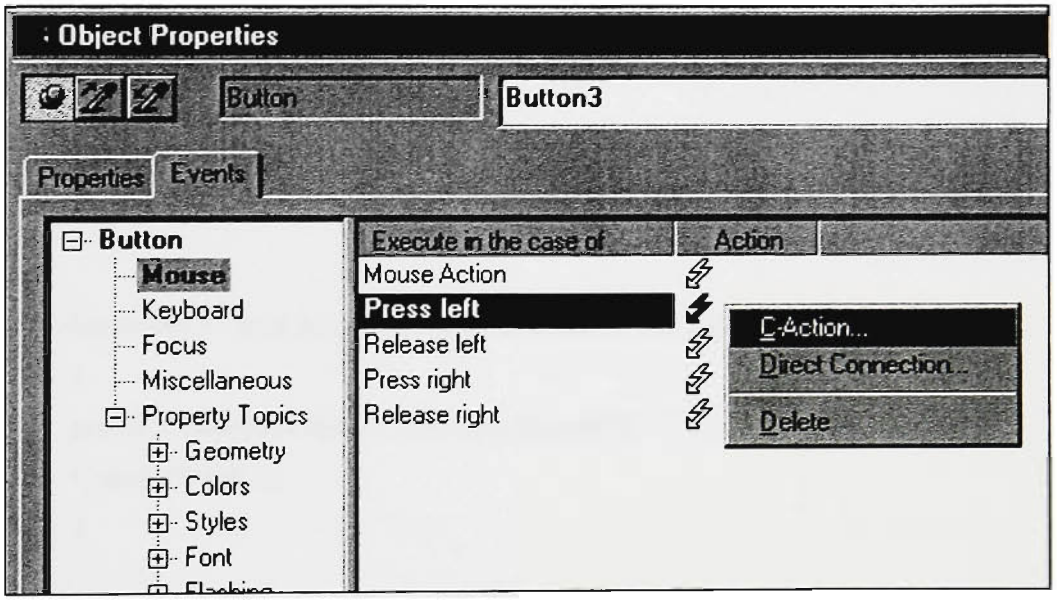


Fig A.13 Object Properties of "Run Query" button of "Order_data" screen

17. Write the following script in the editor window for Run Query button. After writing the script, compile the function by clicking the compile button  on the toolbar. Click OK to save the script.

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName,
UINT nFlags, int x, int y)
{

//*************************************************************************************
**********************************
//This script is executed on "on mouse click" event of Run Query button.
//This script reads data from FoxPro database and writes to WinCC tags
// It connects to the database using the function  "ConnectToDB" and quires the  "order_data" table
//*************************************************************************************
**********************************

//define variables

int iRecordCount;
extern __object *gcn, *grsOrder;


    if (gcn->State==0) // check to make sure the connection was successful
            {
            printf("Connection Failed.  Reconnecting\r\n");
            ConnectToDB ();
            }


                //create an Recordset object that will contain resutls returned by the query
            grsOrder = __object_create("ADODB.RecordSet");

            // using ADODB we populate the recordset object . A dynamic recordset is defined
            // that allows us to navigate through the results and also return the record count.
            grsOrder->CursorLocation = 3;  //to store the cursor on the client, set the cursorlocation 3

/*************************************************************************************
Open method requires 4 parameters "SQL statement", "Name of ADO connection begin used", "Cursor
type", "Lock type")
```

CursorType - Allows you to browse a recordset in different ways: some cursors allow you to move backward and forward in a recordset, while other cursors limit you to moving forward only.If you set the CursorLocation to the client, you must set the CursorType to adOpenStatic (value = 3).

LockType - Determines how (and if) a recordset can be updated. Recordsets can be set to read-only, or they can be configured to allow updates. For most scripts, the LockType can be set to adLockOptimistic (value = 3). With this setting, the record being edited is not locked (that is, no restrictions are placed on another user accessing that record) until you call the Update method.

**********/

```
grsOrder->Open("SELECT Cust_name,Part_name,qty_req,order_numb,date_req FROM order_data",
gcn,3,3);


        iRecordCount = grsOrder->RecordCount(); // will return the number of rows from the
above query
        SetTagWord("NoOfRecords",iRecordCount);


        if(!grsOrder->eof) //Navigate through the recordset
        {
                grsOrder->MoveFirst;
                SetTagChar("cust_name",grsOrder->Fields(0));
                SetTagChar("Part_Name",grsOrder->Fields(1));
                 SetTagByte("qty_req",grsOrder->Fields(2));          //Return-Type :BOOL

                SetTagDouble("Order_Number",grsOrder->Fields(3));//Return-Type :BOOL

                SetTagChar("Req_Date",grsOrder->Fields(4));

                SetTagWord("CurrentRec",1);
                SetTagBit("QueryRan",1);
        }
        else
                MessageBox (NULL, "No Data Returned", "Query Results", MB_OK |
MB_ICONEXCLAMATION | MB_SYSTEMMODAL);


}
```

18. Similarly write the script for Rewind button  on the graphics. After writing the script compile the function by clicking the compile button on the toolbar. Click OK to save the script. The Rewind Button script is as shown below.

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName,
UINT nFlags, int x, int y)
{

//*****************************************************
//This script is executed on "on mouse click" event of Start of file button.
//This script moves to the first record.
//*****************************************************

//define variables
extern __object* grsOrder;

    grsOrder->MoveFirst; //move to first record

    SetTagChar("cust_name",grsOrder->Fields(0));
    SetTagChar("Part_Name",grsOrder->Fields(1));
    SetTagByte("qty_req",grsOrder->Fields(2));
    SetTagChar("Order_Number",grsOrder->Fields(3));  //Return-Type :BOOL
    SetTagChar("Req_Date",grsOrder->Fields(4));

    //set the CurentRec tag to 1
    SetTagDouble("CurrentRec",1);

}
```

19. Write the script for Previous button  on the graphics. After writing the script compile the function by clicking the compile button on the toolbar. Click OK to save the script. The Previous Button script is as shown below.

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName,
UINT nFlags, int x, int y)
{

//*************************************************************************************
//This script is executed on "on mouse click" event of Previous button.
//This script moves recordset pointer to previous record.
//*************************************************************************************

//declare variables
extern __object* grsOrder;
int NoOfRecs, CurrentRec;

NoOfRecs= GetTagWord("NoOfRecords");
CurrentRec= GetTagWord("CurrentRec");

// if the record pointer is at the 1st record then the statments in the if loop will not be executed
if(CurrentRec > 1)

    {
    grsOrder->MovePrevious; // move to  previous record

    SetTagChar("cust_name",grsOrder->Fields(0));
    SetTagChar("Part_Name",grsOrder->Fields(1));
    SetTagByte("qty_req",grsOrder->Fields(2));
    SetTagChar("Order_Number",grsOrder->Fields(3));  //Return-Type :BOOL
    SetTagChar("Req_Date",grsOrder->Fields(4));

    //set the CurentRec tag to 1 less than the current record
    SetTagDouble("CurrentRec",GetTagDouble("CurrentRec")-1);

    }

}
```

20. Write the script for Next button  on the graphics. After writing the script compile the function by clicking the compile button on the toolbar. Click OK to save the script. The Next Button script is as shown below.

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName,
UINT nFlags, int x, int y)
{

//*************************************************************************
//This script is executed on "on mouse click" event of next button.
//This script moves recordset pointer to next record.
//*************************************************************************

//define variables
extern __object* grsOrder;
int NoOfRecs,CurrentRec ;

NoOfRecs = GetTagWord("NoOfRecords");
CurrentRec = GetTagWord("CurrentRec");

// if the record pointer is at the last record then the statments in the if loop will not be executed
if(CurrentRec < NoOfRecs)
        {

        grsOrder->MoveNext; // move to  next record

        SetTagChar("cust_name",grsOrder->Fields(0));
        SetTagChar("Part_Name",grsOrder->Fields(1));
        SetTagByte("qty_req",grsOrder->Fields(2));
        SetTagChar("Order_Number",grsOrder->Fields(3));  //Return-Type :BOOL
        SetTagChar("Req_Date",grsOrder->Fields(4));

        //set the CurentRec tag to 1 more than the current record
        SetTagDouble("CurrentRec",GetTagDouble("CurrentRec")+1);
        }

}
```

21. Write the script for Forward button 

on the graphics. After writing the script compile the function by clicking the compile button on the toolbar. Click OK to save the script. The Forward Button script is as shown below.

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName,
UINT nFlags, int x, int y)
{

//*************************************************************************
//This script is executed on "on mouse click" event of End of file button.
//This script moves to the last record.
//*************************************************************************

//define variables
extern __object* grsOrder;
int NoOfRecs;

NoOfRecs = GetTagWord("NoOfRecords");

        grsOrder->MoveLast;//move to last record

        SetTagChar("cust_name",grsOrder->Fields(0));
        SetTagChar("Part_Name",grsOrder->Fields(1));
        SetTagByte("qty_req",grsOrder->Fields(2));
        SetTagChar("Order_Number",grsOrder->Fields(3));  //Return-Type :BOOL
        SetTagChar("Req_Date",grsOrder->Fields(4));

        //Set the CurentRec tag to "NoOfRecs"
        //"NoOfRecs" it contains the number of records data
        SetTagDouble("CurrentRec",NoOfRecs);

}
```

22. Write the script for Clear button on the graphics. After writing the script compile the function by clicking the compile button on the toolbar. Click OK to save the script. The Clear Button script is as shown below.

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName,
UINT nFlags, int x, int y)
{
//*************************************************
//This event is trigged on mouse click of Clear button
//This script clears the data value of the tags.
//*************************************************

//define variables
extern __object *grsOrder;

        grsOrder->Close;
        __object_delete(grsOrder); //delet the "grsOrder" object

        SetTagWord("NoOfRecords",0);
        SetTagWord("CurrentRec",0);
        SetTagChar("cust_name","");
        SetTagChar("Part_Name","");
        SetTagByte("qty_req",0);
        SetTagChar("Order_Number",""); //Return-Type :BOOL
        SetTagChar("Req_Date","");
        SetTagBit("QueryRan",0); //set the QueryRan tag to 0

        printf("closed recordset \r\n");

}
```

23. The Exit button deactivates the project, the code for which is as follows:

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName,
UINT nFlags, int x, int y)
{
 DeactivateRTProject ();
}
```

24. Run the WinCC Graphic and click Run Query. If there are any problems with the script then it will be seen in the "Global script – Diagnostic" window.

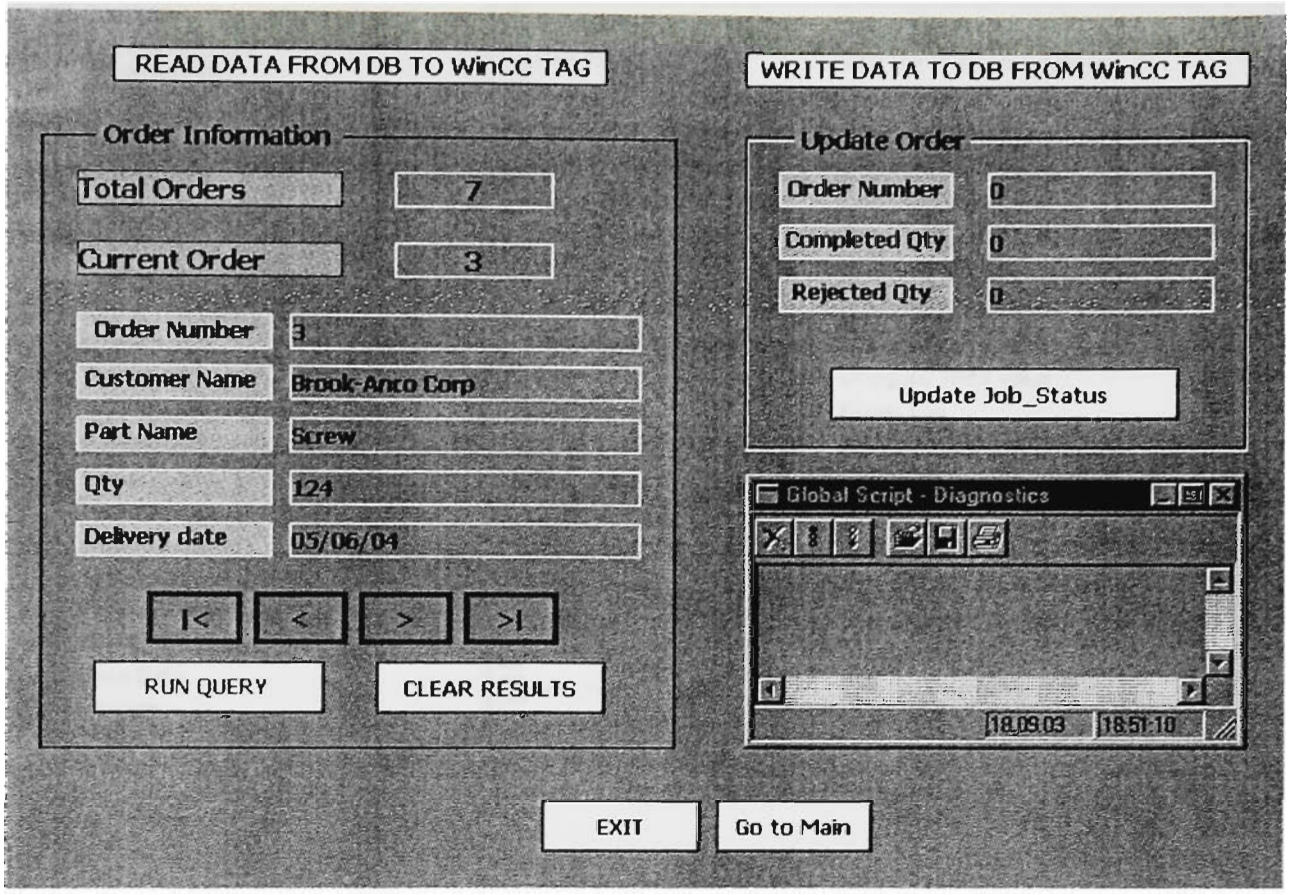
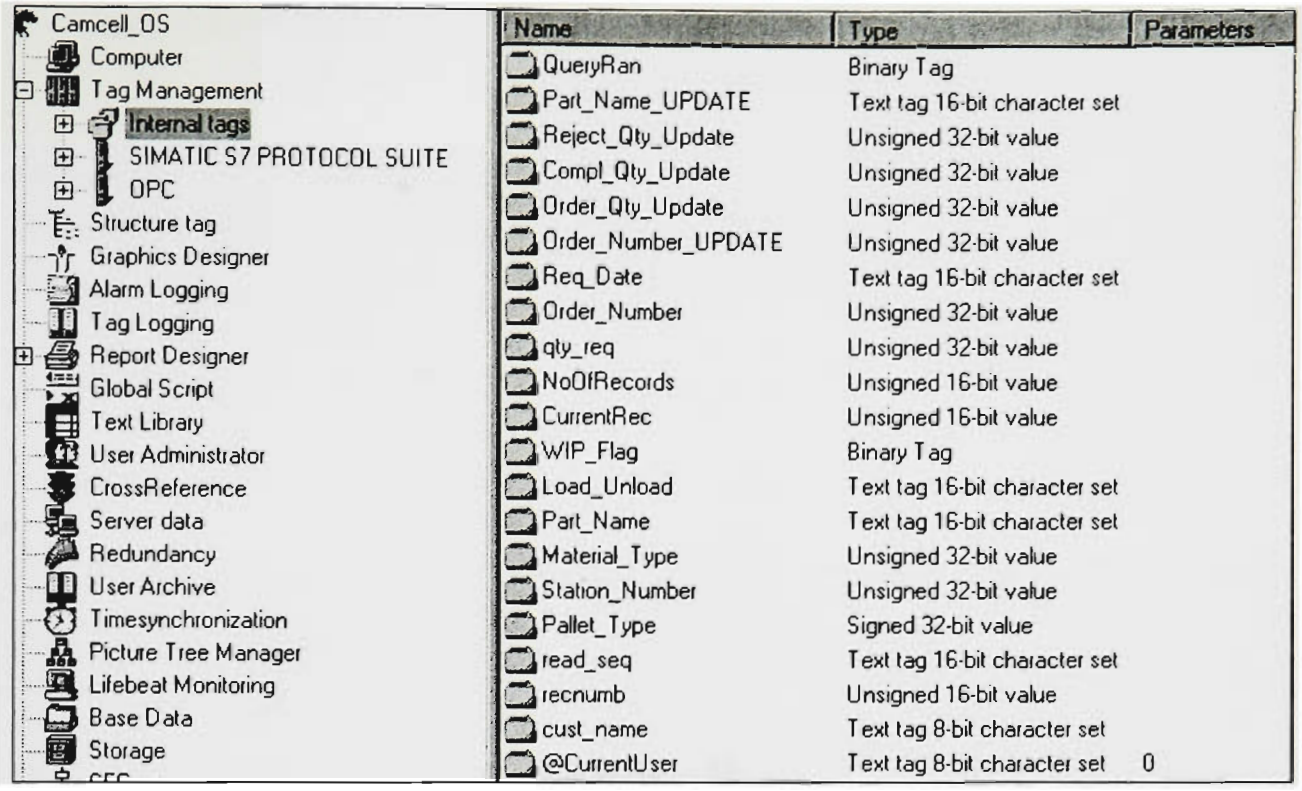


Fig A.14 Order_data screen

25. Go back to Main.pdl and configure the "READ DATA FROM DB TO WinCC TAG" button so that during runtime it will take the operator to "Order_Data.pdl" screen.

26. We have completed the procedure to read data from database and transfer it to WinCC graphics via WinCC tags.

Attachment 1



Appendix B

Write Data from WinCC tags to FoxPro database

Appendix B. Write Data from WinCC tags to FoxPro database

Purpose

The purpose of this document is to give step-by-step instructions to write the data from WinCC tags to FoxPro database. This will involve updating Job status information for an order.

Pre-requisite

The database and WinCC graphics should be completed as described in Appendix A. WinCC tags must be created as per the list provide in Attachment 1 of appendix A.

Procedure:

1. Open the "Order_data.pdl" graphics.

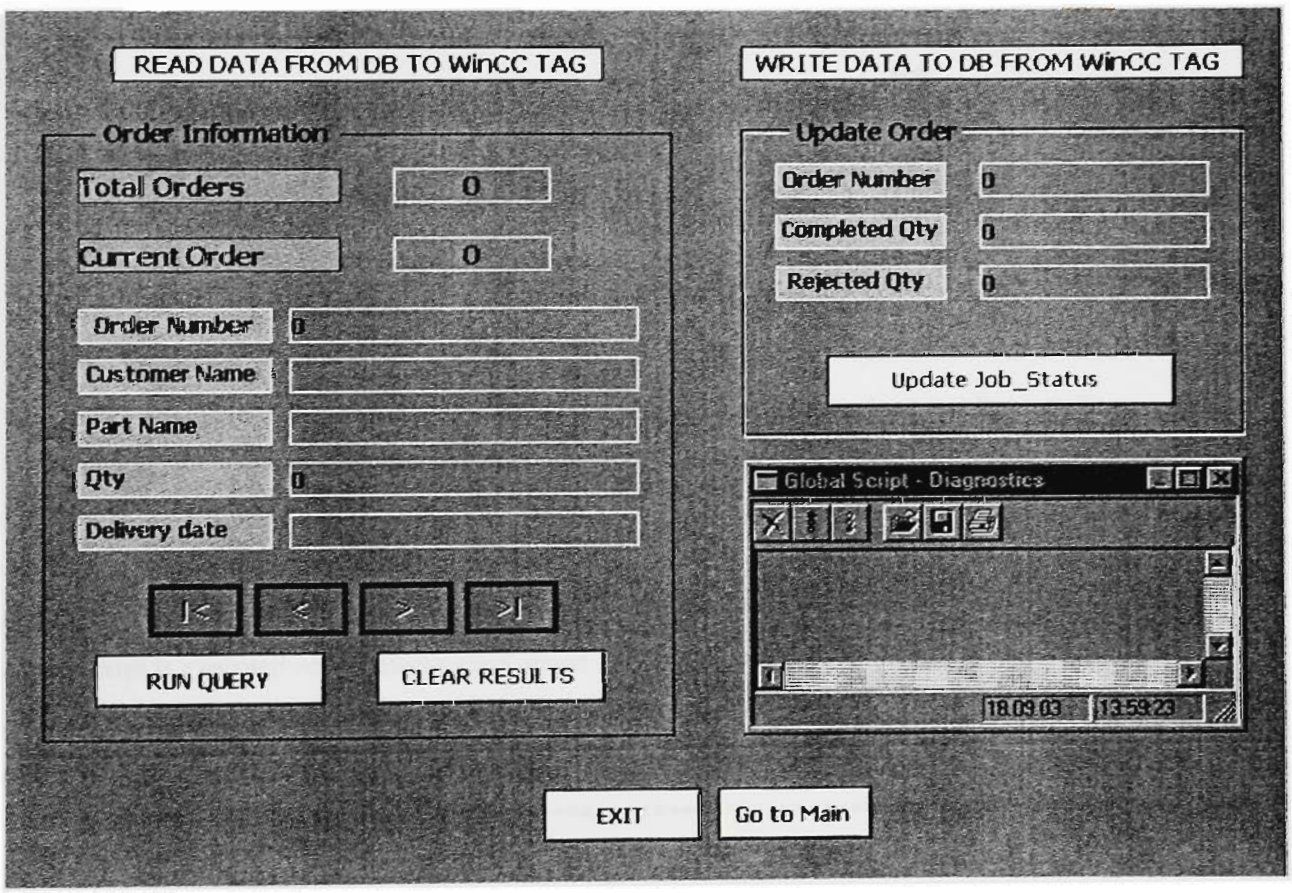


Fig B.1 Order_data screen

2. Next, we write the C script in the mouse click event of "Update Job Status" Button. Double click on the "Update Job Status" button. This will open the property window of the button as shown in Fig. B.2. In Events select "Mouse Press left" and right click on the Action. Select C-Action option from the pop up menu.

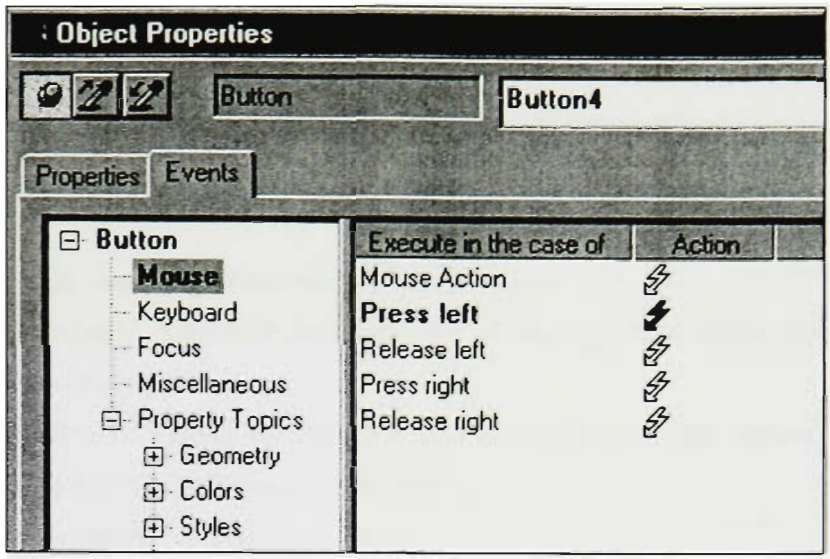


Fig B.2 Object properties of "Update Job Status" button

3. Write the following script in the editor window for Update Job Status button. After you are done writing the script compile the function by clicking the compile button on the toolbar. Click OK to save the script.

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName,
UINT nFlags, int x, int y)
{

//*************************************************************************
//This script is executed on the mouse click event of "Update Job_Staus" button
//This script writes the data from WinCC tags to FoxPro database
// It connects to the database using the function  "ConnectToDB" and updates the  "Job_status" table
//*************************************************************************

//define variables

char sQuery[900];
extern __object *gcn, *grsOrderUpdate;


    if (gcn->State==0) // check to make sure the connection was successful
            {
            printf("Connection Failed\r\n");
            ConnectToDB (); //Return-Type :void
```

```
        }

        //create a Recordset object that will contain results returned by the query
        grsOrderUpdate= __object_create("ADODB.RecordSet");

        //the SQL statment updates the "Job_status" table.
        sprintf(sQuery, "UPDATE Job_Status SET Compl_qty='%s', Reject_qty='%s' WHERE
        Order_numb='%s'",
        GetTagChar("Compl_Qty_Update") ,GetTagChar("Reject_Qty_Update"),
        GetTagChar("Order_Number_UPDATE"));
        grsOrderUpdate->Open(sQuery, gcn,1);

        //once the data is updated the a message box will pop up confirming the update.
        MessageBox (NULL, "Update Job_status table", "UPDATE", MB_OK |
MB_ICONEXCLAMATION | MB_SYSTEMMODAL);

        __object_delete(grsOrderUpdate); //destroy the "grsOrderUpdate" object

}
```

4. Run the WinCC Graphics and click Run Query this will enable the "Update Job_status" button. If there are any problems with the script then it will be seen in the "Global script – Diagnostic" window.
5. Enter a sample data in "order number", "completed_qty" and "rejected_qty". On clicking update button a message will pop up confirming that the order has been updated.

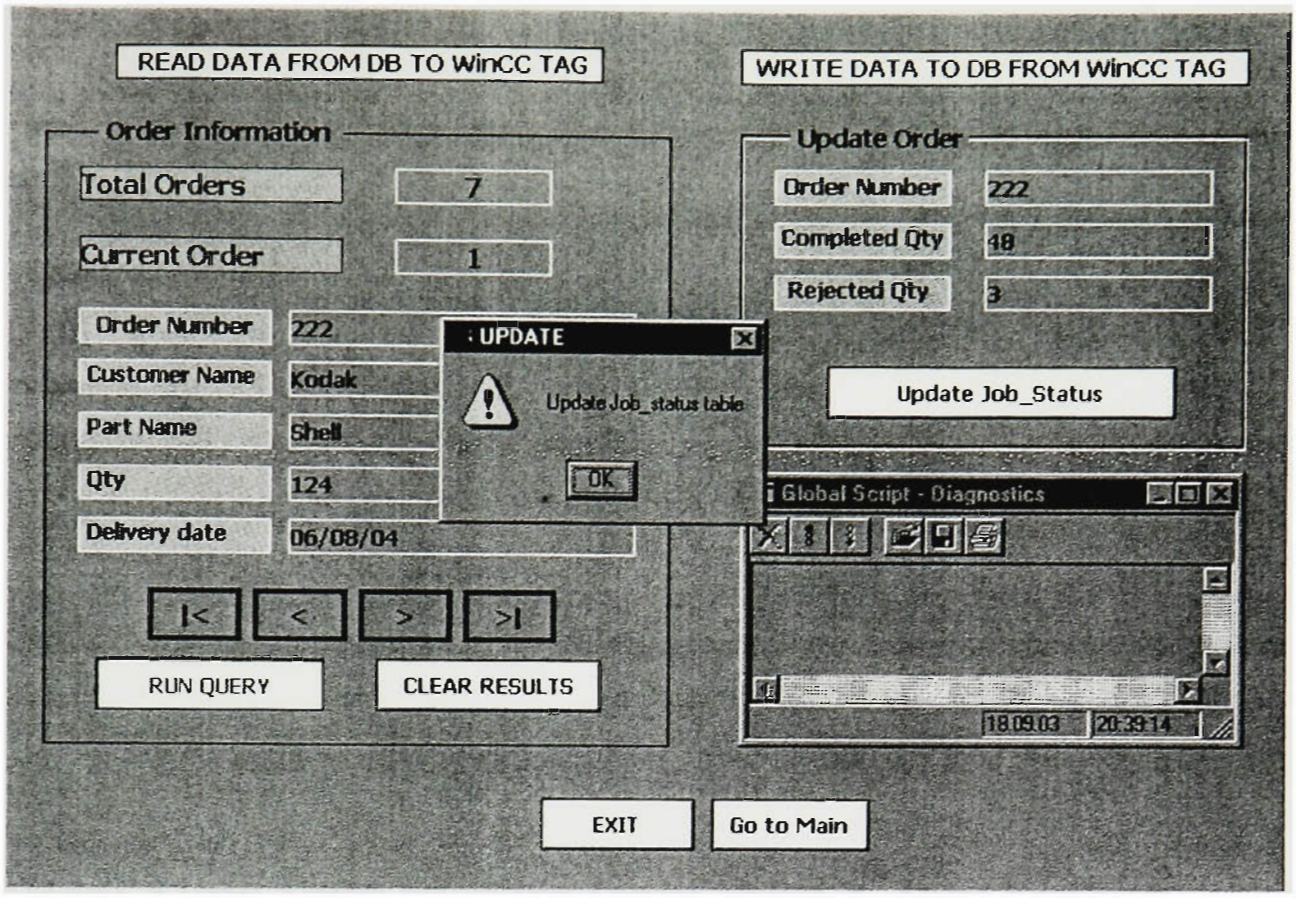


Fig B.3 Update message box

6. Open the job_status table of Order database to confirm the updated data.

Job_status

Order_numb	Part_name	Total_order_qty	Compl_qty	Reject_qty
222	Shell	124	48	3
225	TMale	111	12	0
220	Bolt	200	123	2

Fig B.4 Job_status table

7. The procedure to write data from WinCC tag to database is now complete.

Appendix C

Real Time Data capture from WinCC into FoxPro Database

Appendix C. Real Time Data capture from WinCC into FoxPro Database

Purpose
This purpose of this document is to give step-by-step instructions to capture Real Time Step 7 & WinCC data into FoxPro database.

Pre-requisite
This tutorial uses the "Camcell" Step 7 Program and "Camcell_OS" WinCC application that have already been created for the CAMCELL system. These applications must be installed prior to developing this tutorial. Real-time data will be captured for Pallet count at every station. The following Step 7 tags keep track of the pallet count:

- S7$Program(1)/Pallet$CountAtVision
- S7$Program(1)/Pallet$CountAtMill
- S7$Program(1)/Pallet$CountAtLathe
- S7$Program(1)/Pallet$CountAtDock
- S7$Program(1)/Pallet$CountAtAsembly1
- S7$Program(1)/Pallet$CountAtAsembly2
- S7$Program(1)/Pallet$CountAtAsembly3
- S7$Program(1)/Pallet$CountAtAsembly4

Also User Archive Editor and FoxPro database must be installed.

Procedure:
The procedure for collecting real-time data from the WinCC can split into 3 main steps:

I. Define tags in User Archive Editor.
II. Write script to update User Archive WinCC tags with real-time data.
III. Create connection between WinCC runtime Sybase database and FoxPro database.

1. We start off by creating a ***New User Archive***. Open the User Archive editor from the WinCC Control Center. To do so right click on the User Archive icon and select the Open option form the pop up menu.

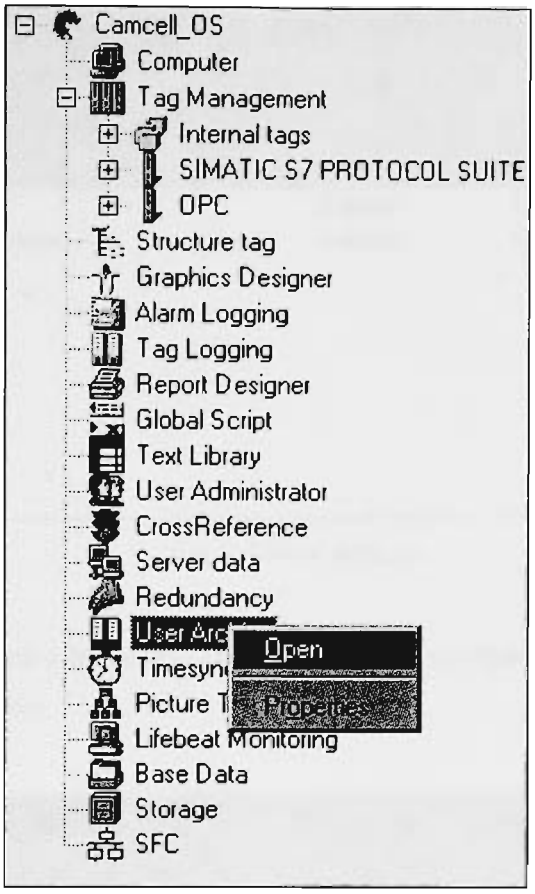


Fig C.1 WinCC navigator

2. A project window will open as shown below. The Project window is divided in to three windows viz. the "navigation window"," data window" and "table window"

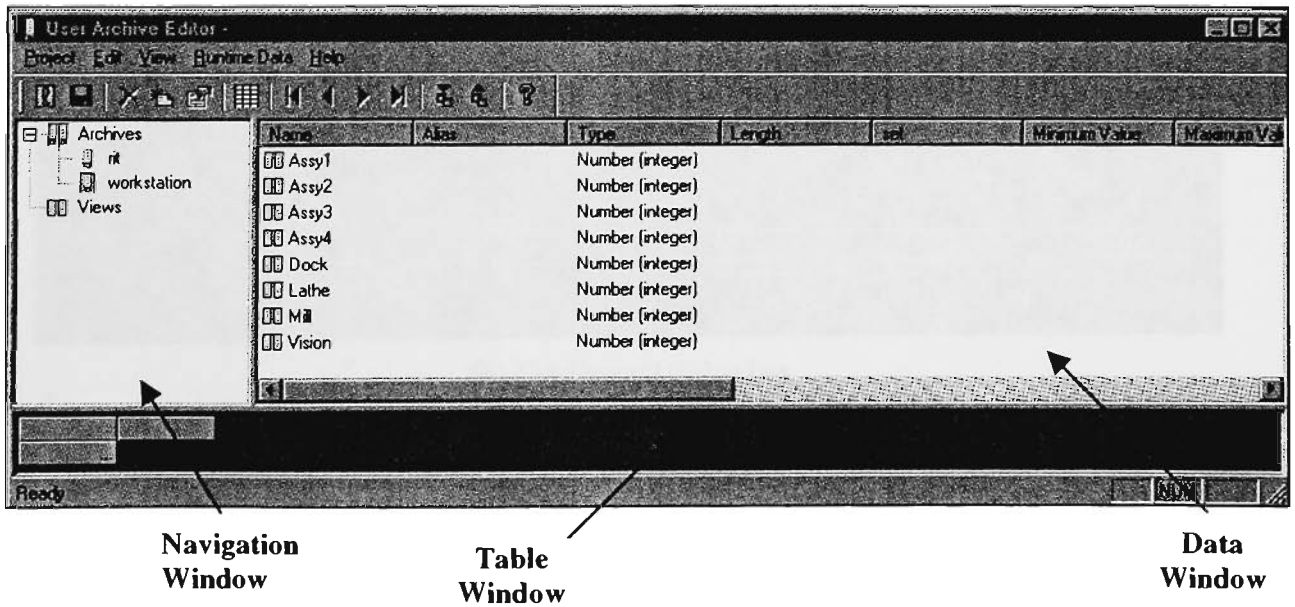


Fig C.2 User Archive editor

3. In the navigation window, click on "Archives". Right click on the navigation or data window. The following pop-up menu will be displayed.

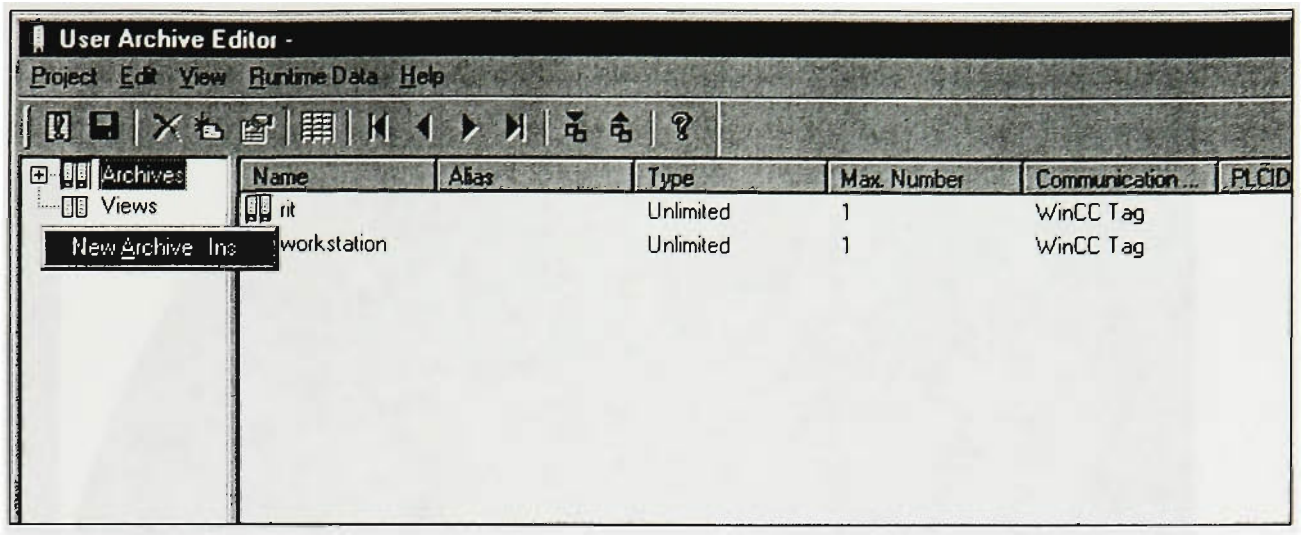


Fig C.3 New Archive

4. Click on the "New Archive Ins" entry. The Wizard for the configuration of archives will be displayed as shown below.

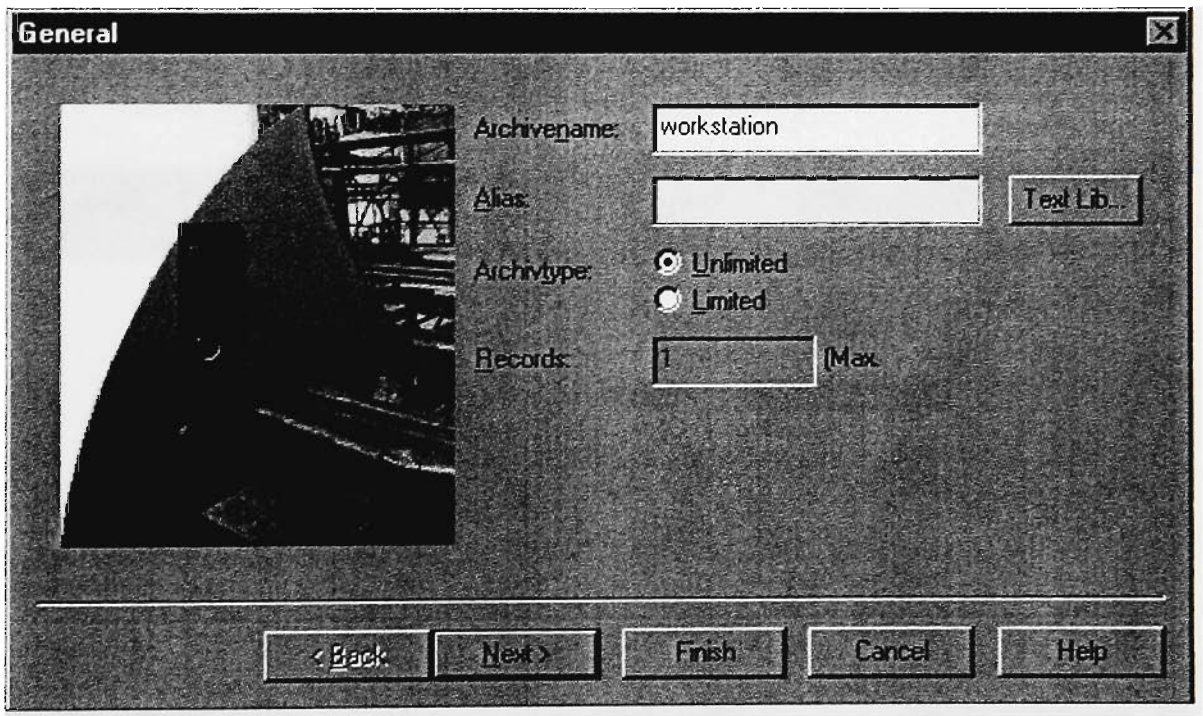


Fig C.4 General dialog box

5. Give a suitable Archivename to the archive. The archive in this particular application is named as "workstation". Make sure not to use keywords of SQL language. Click next button.
6. In the "Communication" dialog box, the connection type between the PLC and the archive is set: In the "Type" area, the communication type can be defined:
    - None: No communication possible
    - Via Raw Data Tag: Access to PLC via a raw data tag
    - Via WinCC Tag: Access to PLC via a WinCC tag

    Since we will "communicate via a WinCC tag" select the third option and click OK.

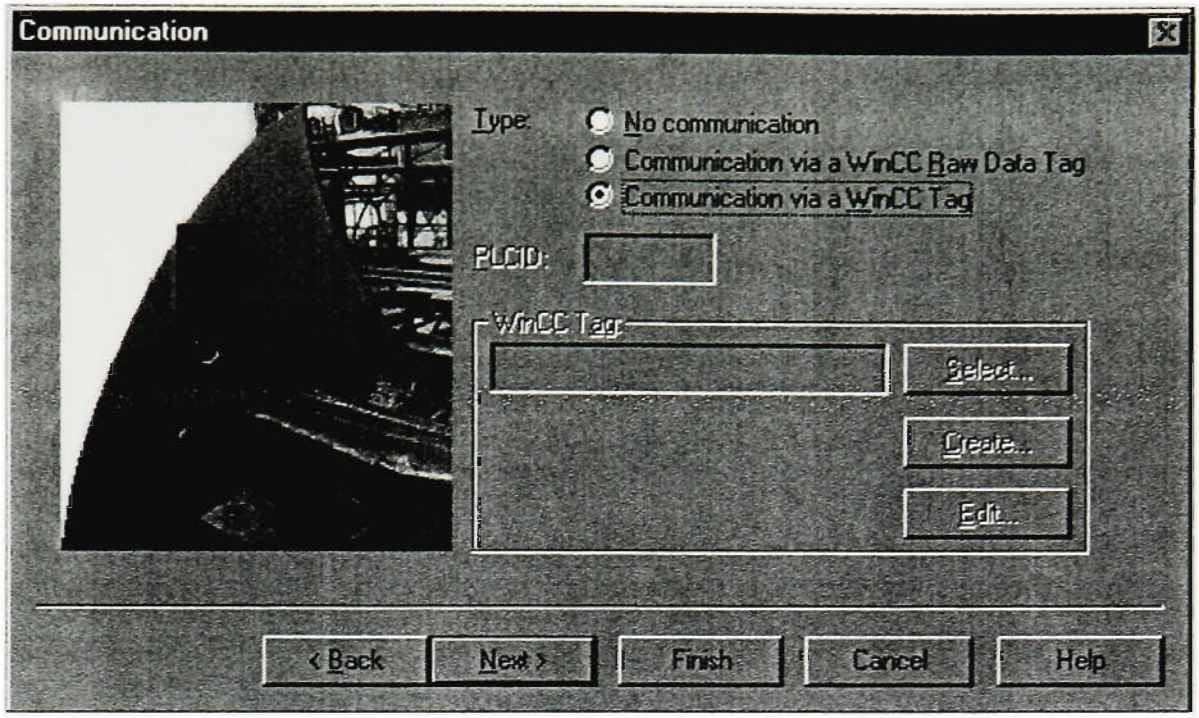


Fig C.5 Communication Dialog box

7. The "Control Tab" dialog opens as shown in the figure below

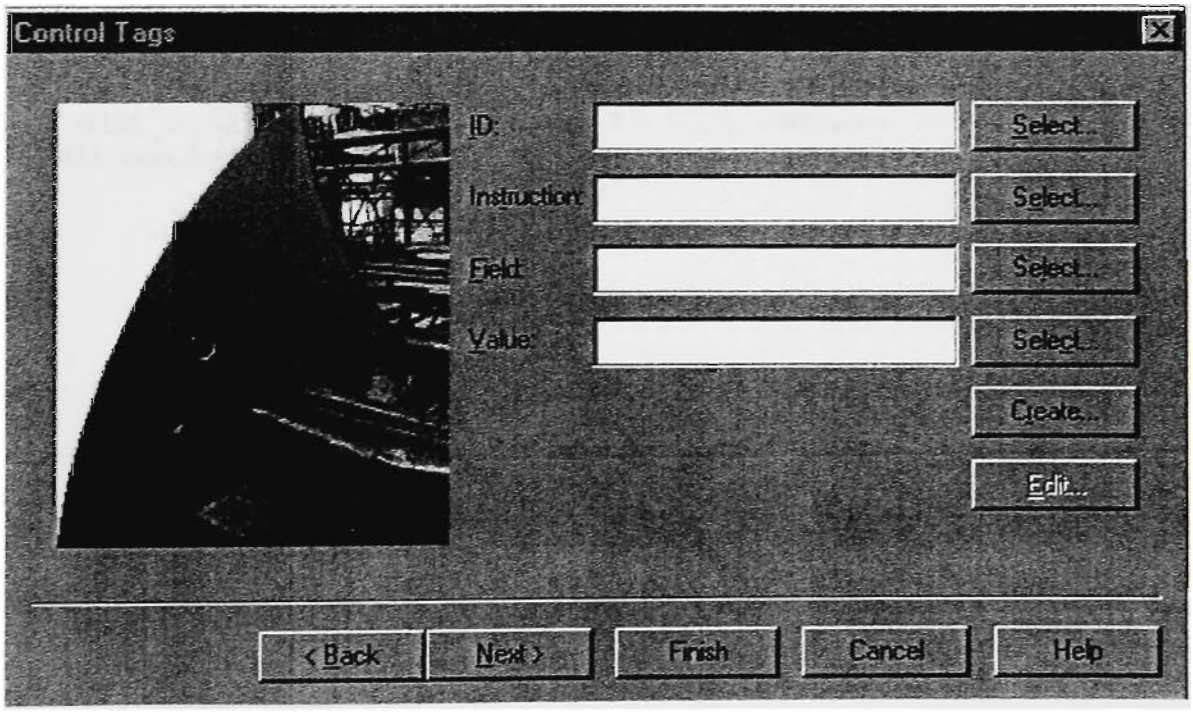


Fig C.6 Select control tags

8. In the "Control Tags" tab, control tags in the form of WinCC tags are set up that are used to access archive fields. In order to generate new tags automatically click on Create button.

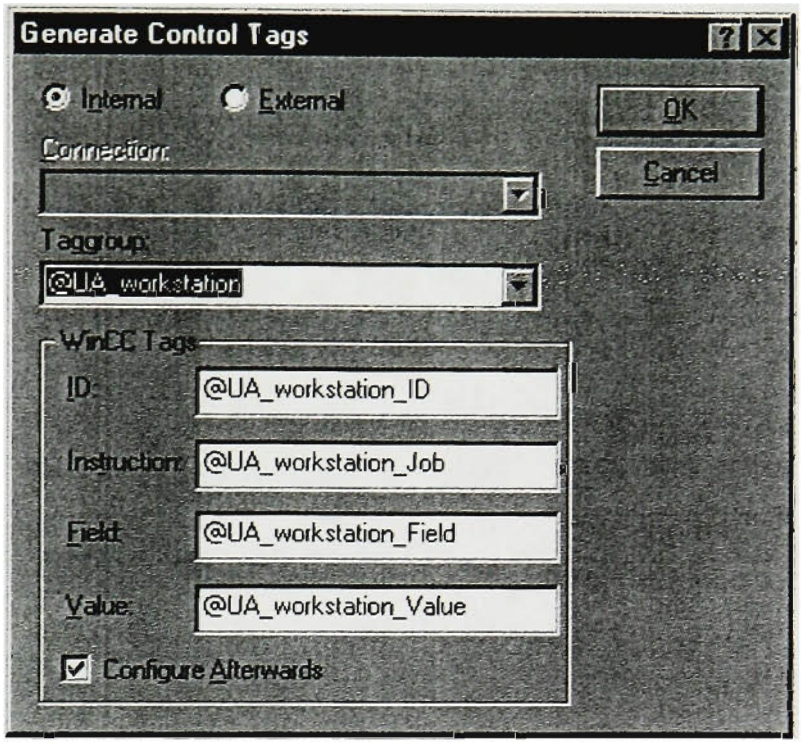


Fig C.7 Generate Control tags dialog box

This will set up a new tag group "@UA_workstation" consisting of the generated tags in the form of @UA_ workstation ID, @UA_workstation _Job, @UA_ workstation _Field and @UA_workstation_Value. Click OK button.

9. The Tag Properties window as shown below will appear for all the four tags. In our case the datatype for all tag is "Signed 32-bit value" Click OK for all.

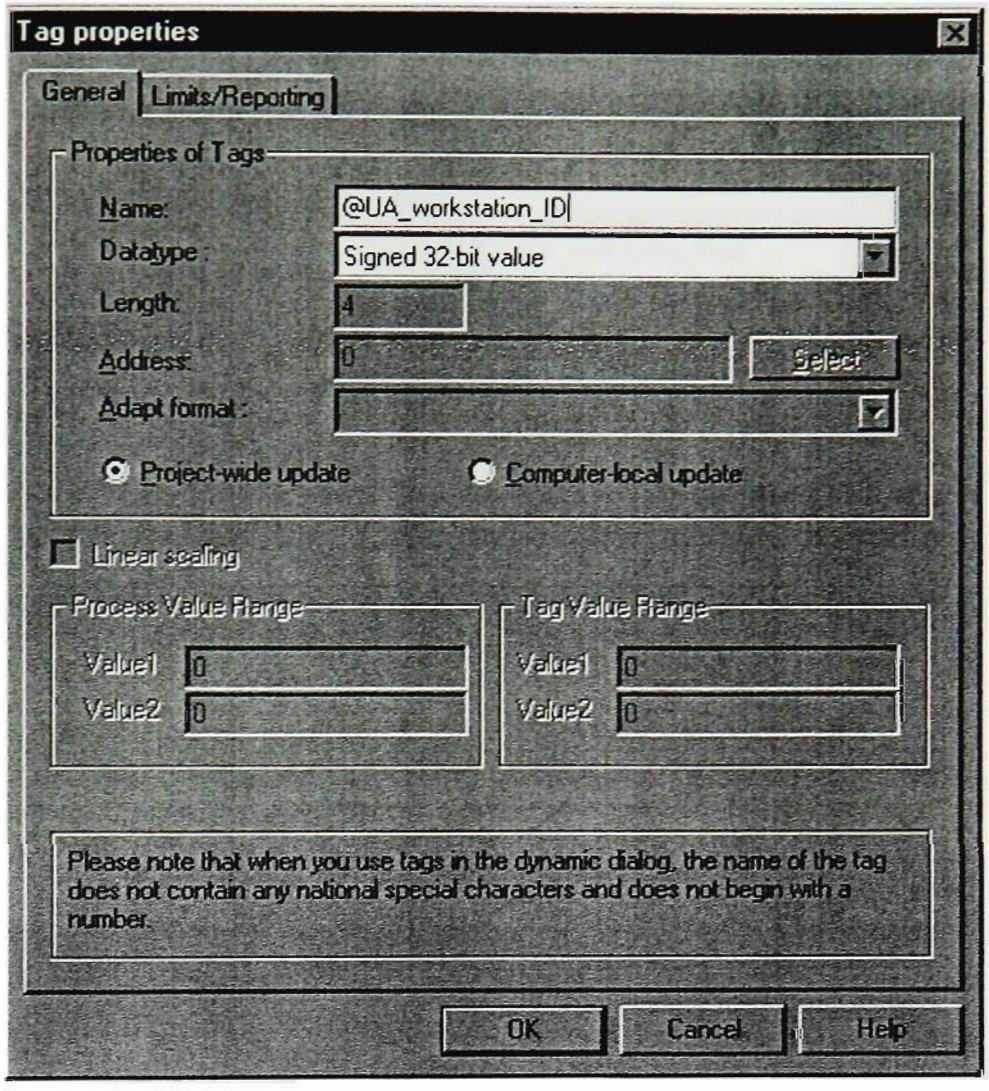


Fig C.8 Tag properties dialog box

10. A dialog box as shown below opens click OK to proceed.

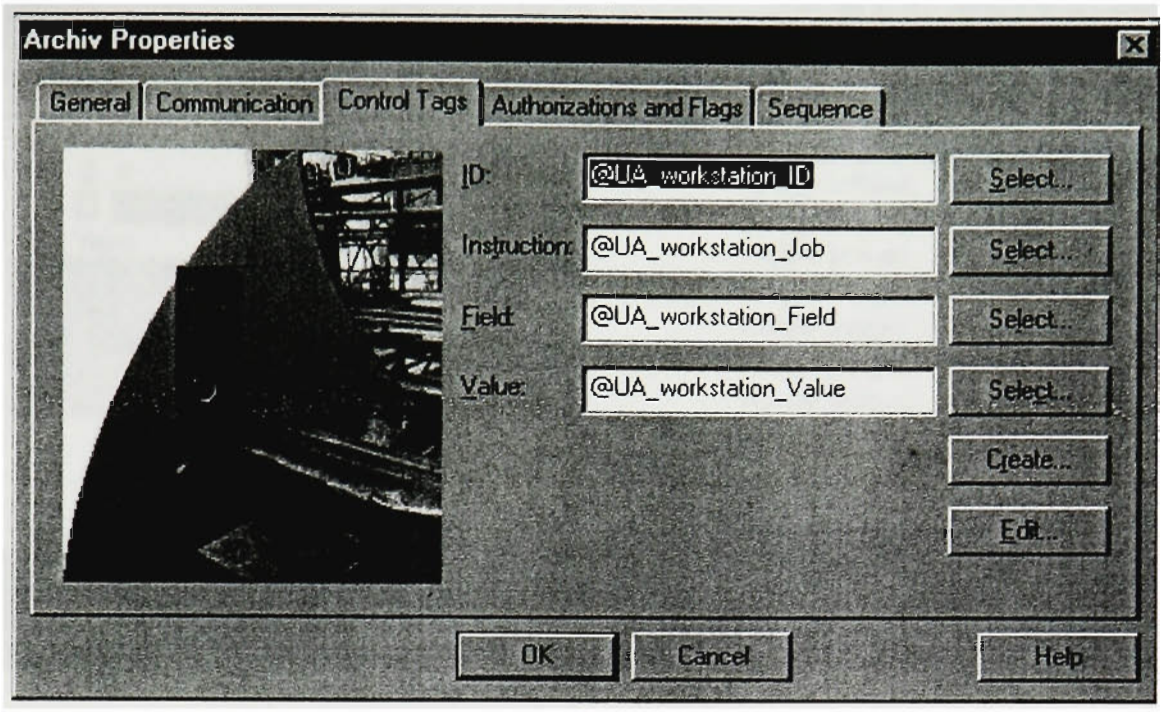


Fig C.9 Archive Properties dialog box

11. Authorization dialog box opens where one can specify access rights and properties of the archive. We don't need to assign Authorization. Click Finish.

    A New User Archive has been created. The next step is to assign the tags that need to be archived.

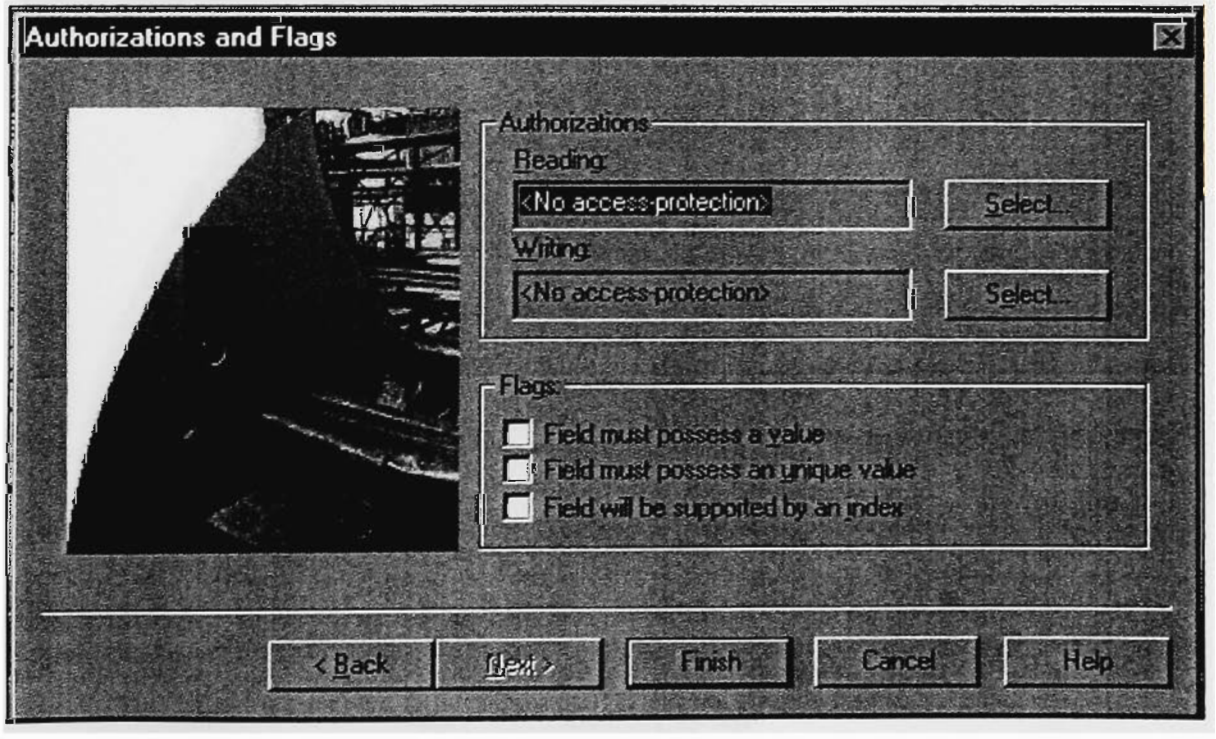


Fig C.10 Authorization and Flags dialog box

12. To create **New Archive fields**, select the archive name "workstation", right click on it and select the option New Field from the pop up menu.

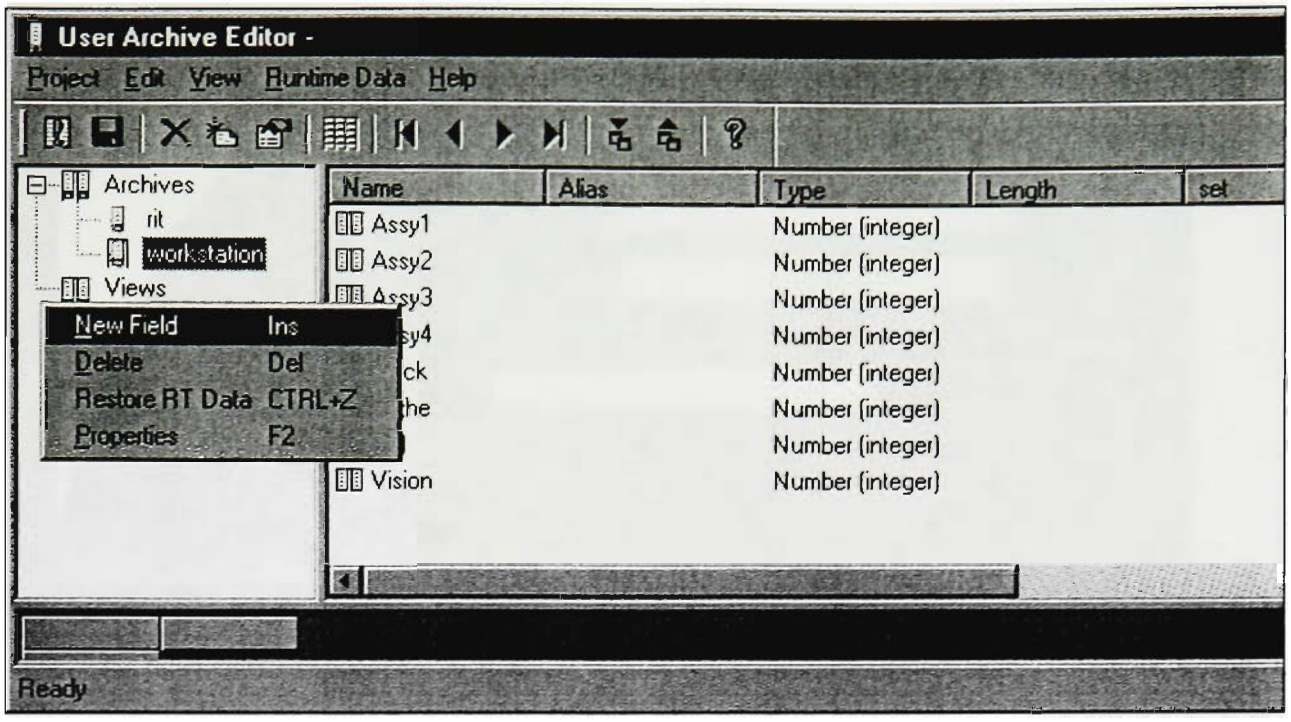


Fig C.11 Generate New Field

13. The "General" dialog box opens as shown below. Specify the archive field name that one wants to create as well as the field type. In this application Fieldname is "Vision" of Integer data type. Click next to continue.

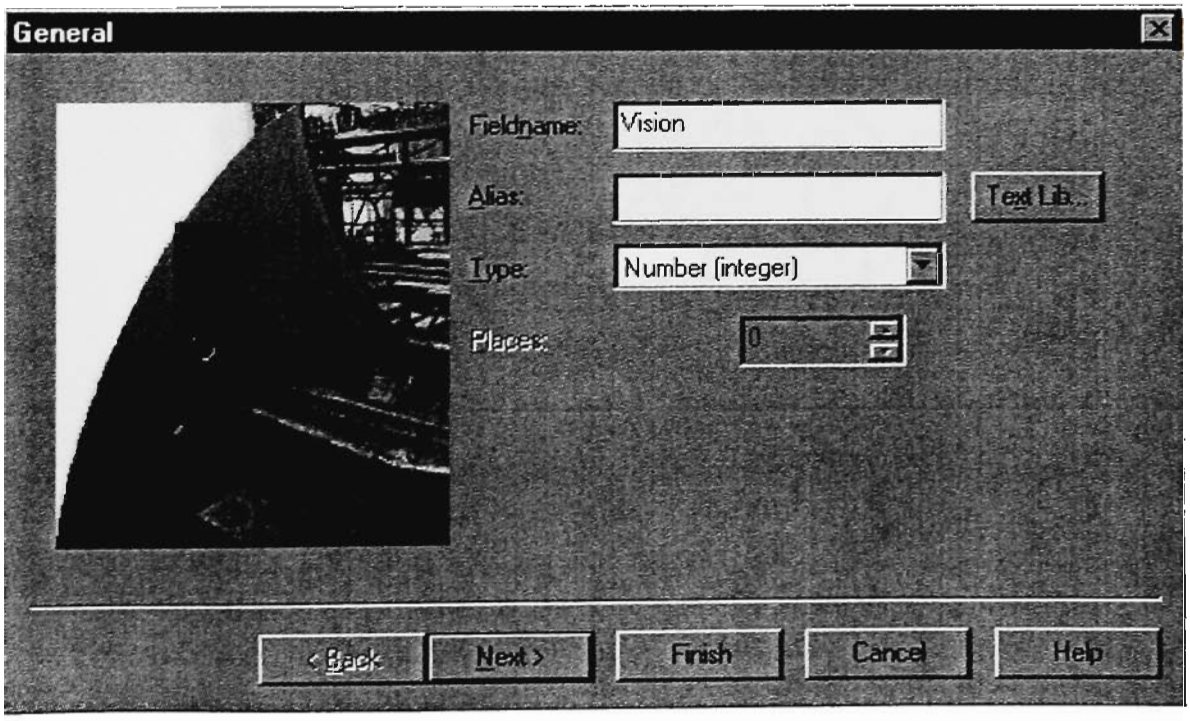


Fig C.12 General dialog box

14. A dialog box as shown below opens. We will create a new WinCC Tag automatically. Click on Create button.

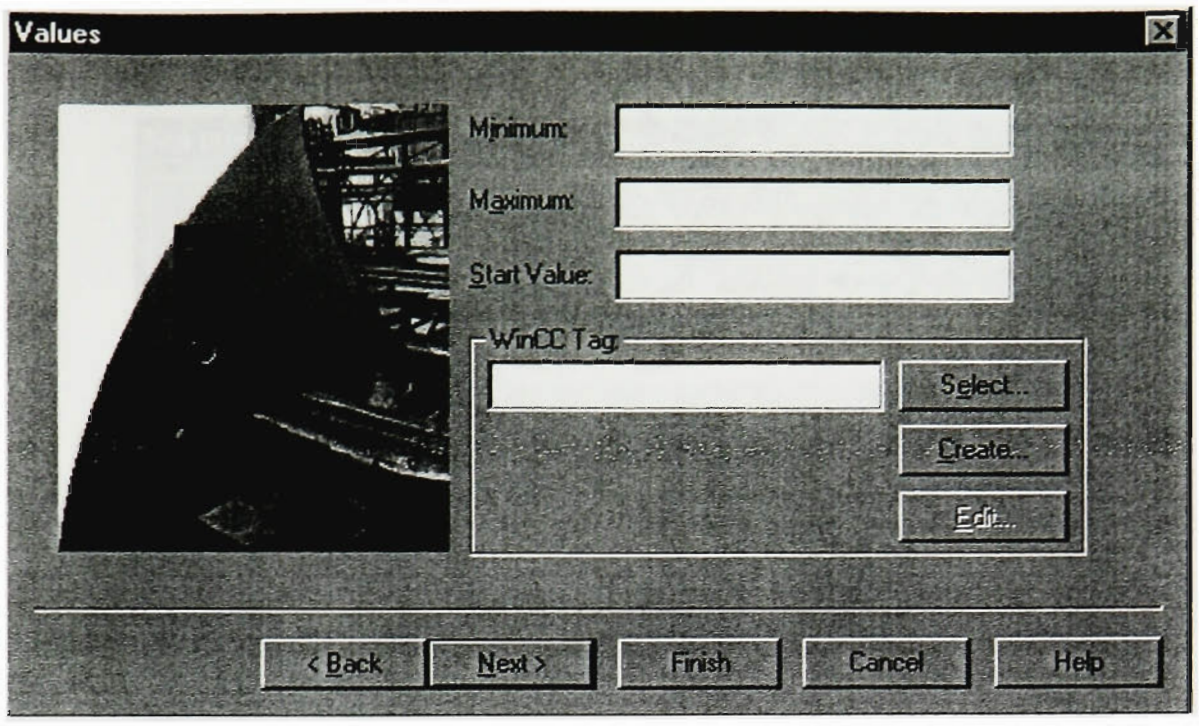


Fig C.13 Value dialog box

15. In the create WinCC tag dialog box select the Tag group "@UA_workstation". It automatically names the Field name as "@UA_workstation_Vision" Click OK to continue.

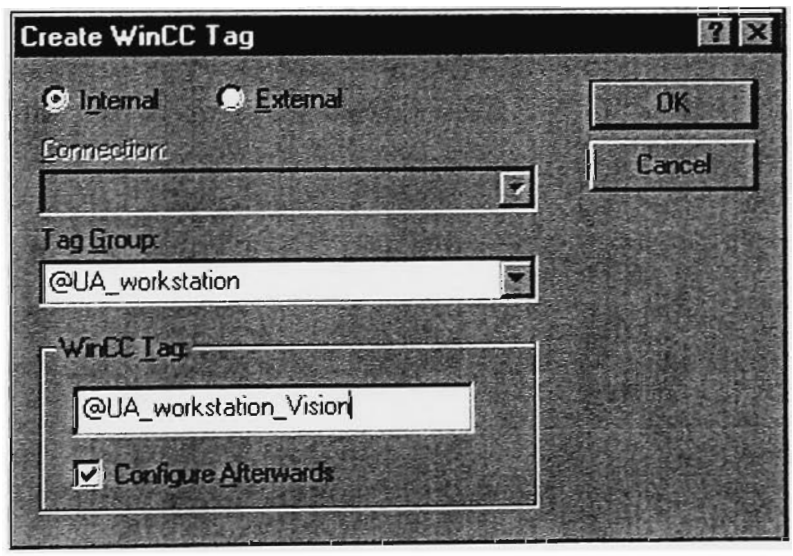


Fig C.14 Create WinCC tag dialog box

16. The Tag Properties window as shown below will appear. In our case the datatype for "@UA_workstation_Vision" is "Signed 32-bit value". Click OK.

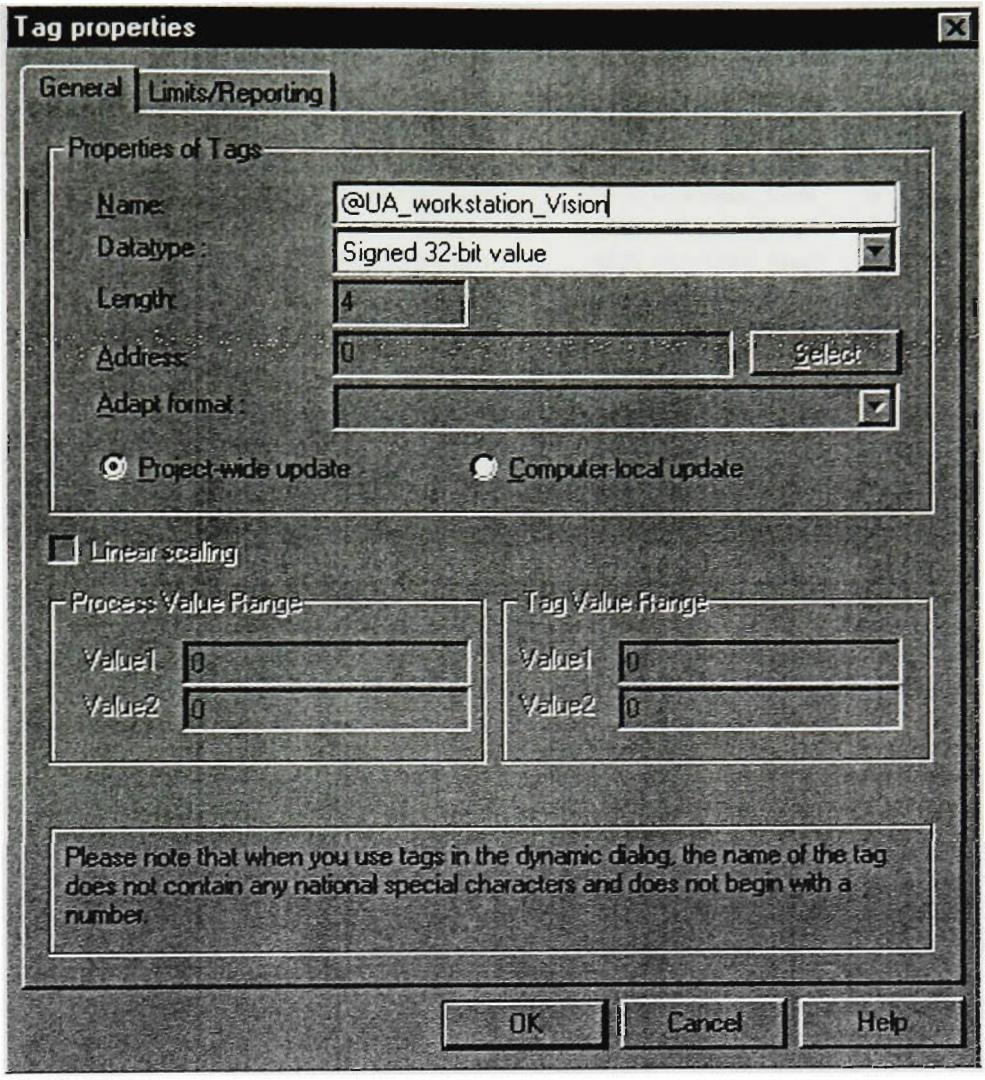


Fig C.15 Tag properties dialog box

17. Click Finish. We don't need to assign Authorization for Archive fields.

Repeat Steps 12 to 17 to create remaining Archive fields upon which the User Archive Editor should look similar to the one shown below.

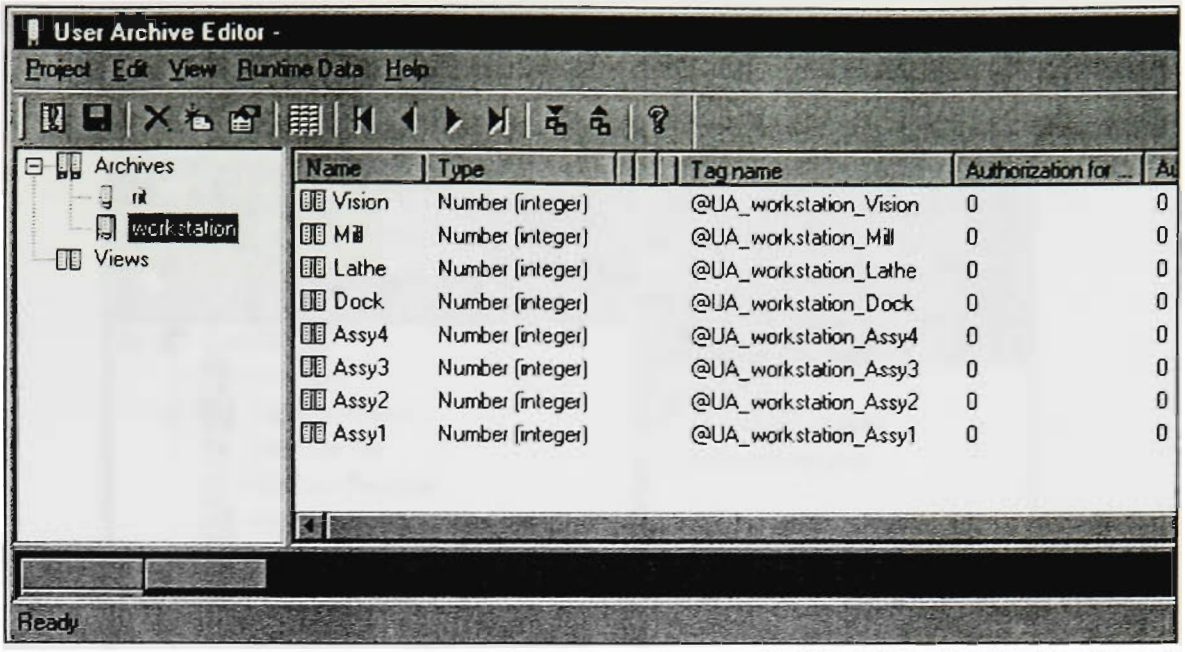


Fig C.16 User Archive editor

18. In the WinCC navigation window expand the "Tag management" option in the Internal tags option and search for the tag group name "@UA_workstation" This tag group contains all the tags that were created using User Archive.

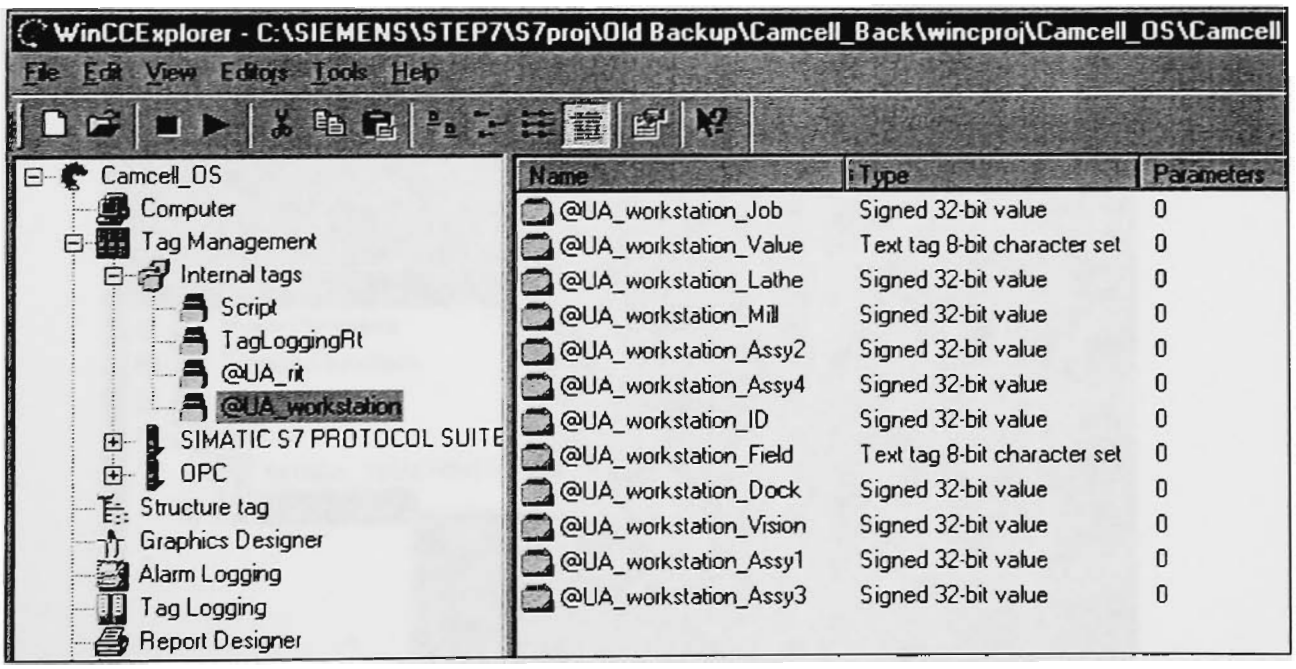


Fig C.17 Tag management

The creation of the User Archive (UA) WinCC tags is complete .For each UA WinCC tag there exists a corresponding Step 7 tag. These Step 7 tags represent the field device and contain the real time data. The UA WinCC tag has to continuously poll the Step 7 tag in order get the real time data. This can be achieved by **Global Action** and **Trigger** functions of Global Script Editor. The Global Action will contain the script for setting the value of Step 7 tag to UA WinCC tag. The Trigger will be linked with the action and will initiate an event for calling the action and triggering it. Below are the steps to create the Global Action.

19. From the WinCC Control Center open the Global Script editor. To do so right click on the Global Script icon and select the Open option form the pop up menu.

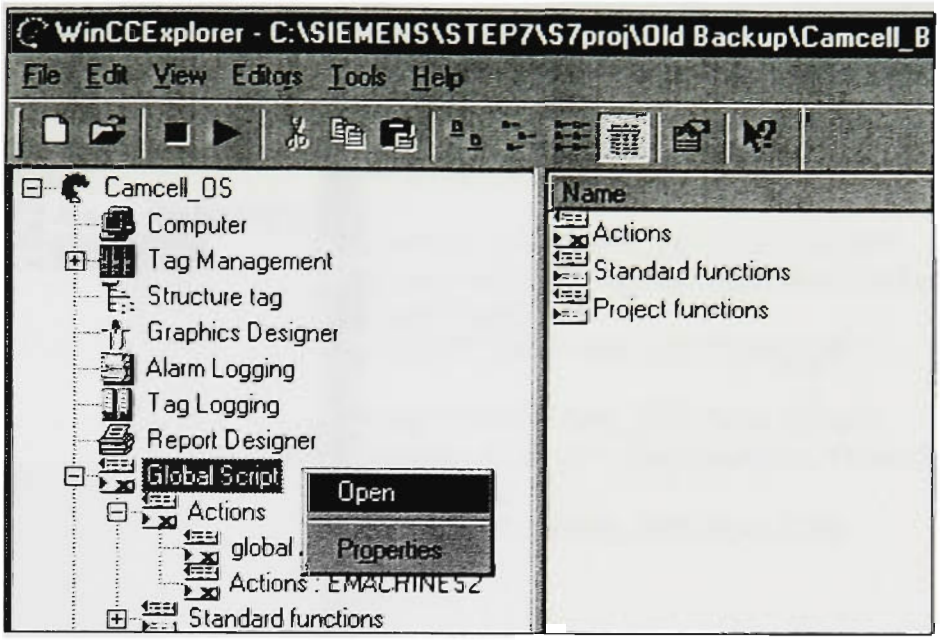


Fig C.18 Global Script

20. Expand the "Actions" option and select "global Actions" from the menu. Create a new "global Action"

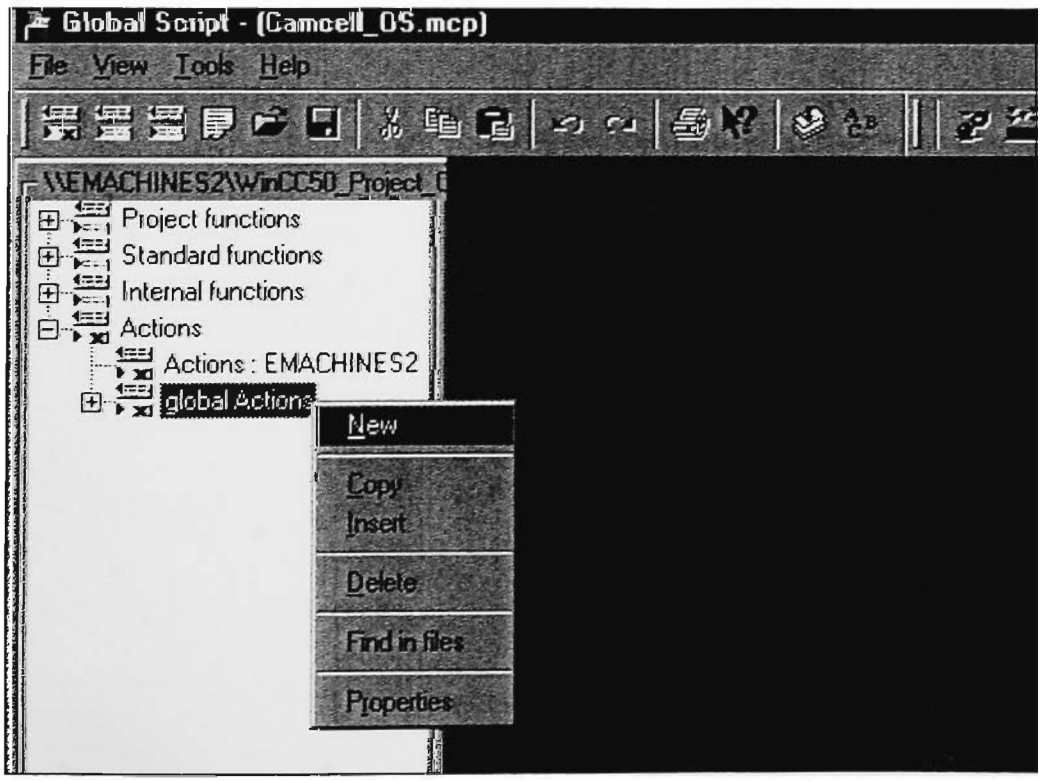


Fig C.19 Global action

21. A new action editor window will open. The "gscAction" is a global action function that is a name defined by the system and cannot be changed.

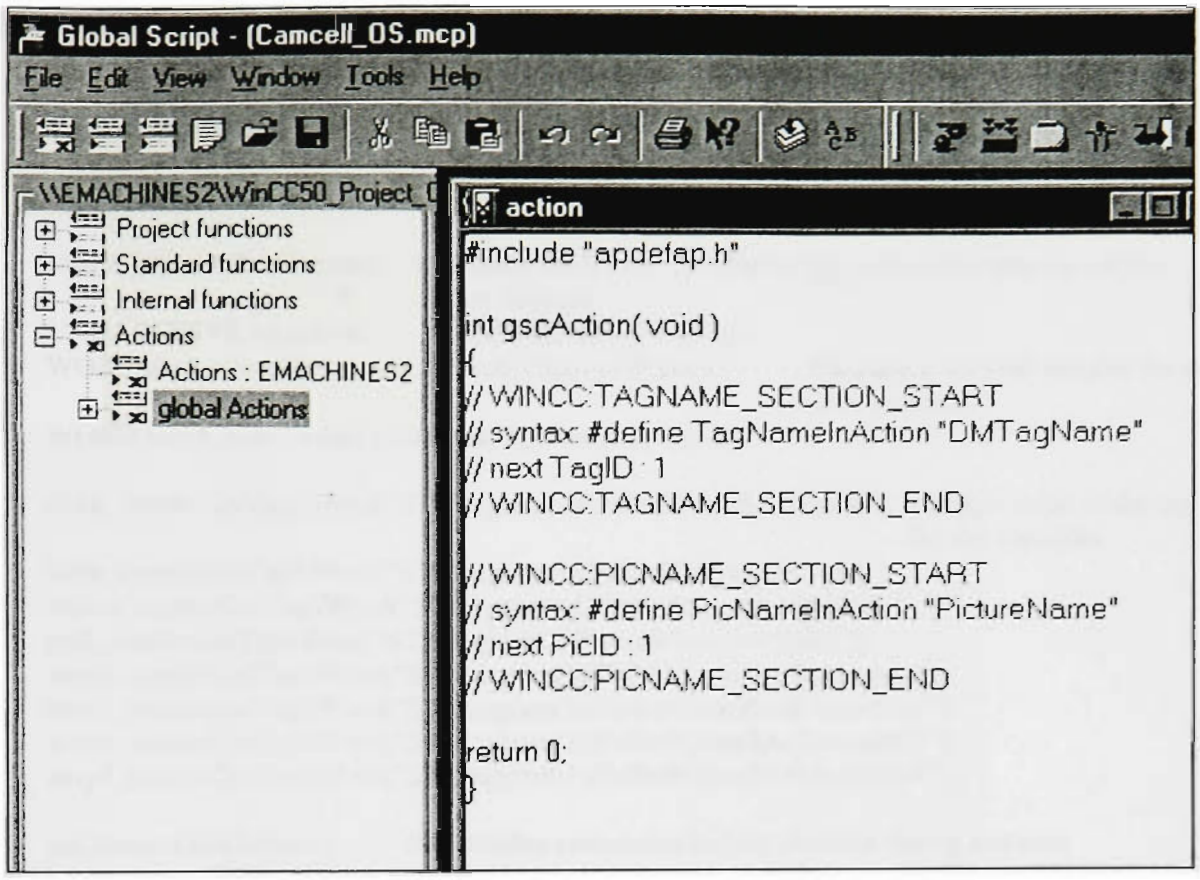


Fig C.20 New Action

22. Write the script shown below in the editor.

```
#include "apdefap.h"

int gscAction( void )
{
UAHCONNECT hConnect;    //defining the handle variable for the component that we call for
                        //User Archive
UAHARCHIVE hArchive;    //handle for specific archive
WORD dock_count,lathe_count,vision_count,mill_count;        //declare a internal variable for all
                                                            //the counters
WORD assy1_count,assy2_count,assy3_count,assy4_count;

dock_count=GetTagSWord("S7$Program(1)/Pallet$Count$At$Dock"); //assign value of the tag
                                                              //to the variables
lathe_count=GetTagSWord("S7$Program(1)/Pallet$Count$At$Lathe");
vision_count=GetTagSWord("S7$Program(1)/Pallet$Count$At$Vision");
mill_count=GetTagSWord("S7$Program(1)/Pallet$Count$At$Mill");
assy1_count=GetTagSWord("S7$Program(1)/Pallet$Count$At$Asembly1");
assy2_count=GetTagSWord("S7$Program(1)/Pallet$Count$At$Asembly2");
assy3_count=GetTagSWord("S7$Program(1)/Pallet$Count$At$Asembly3");
assy4_count=GetTagSWord("S7$Program(1)/Pallet$Count$At$Asembly4");

uaConnect(&hConnect);       //establishes connection to User Archive during run time

if (uaQueryArchiveByName(hConnect,"workstation",&hArchive)==FALSE);   //connect the
                                                                      //archive via name "workstation"
{
printf("problem with query/r/n");
}

uaArchiveOpen(hArchive);                    //open the archive

SetTagDWord("@UA_workstation_ID",1);              // write the value to 1st record
SetTagDWord("@UA_workstation_Dock",dock_count);   //write the value to respective
                                                  //control tag
SetTagDWord("@UA_workstation_Lathe",lathe_count);
SetTagDWord("@UA_workstation_Vision",vision_count);
SetTagDWord("@UA_workstation_Mill",mill_count);
SetTagDWord("@UA_workstation_Assy1",assy1_count);
SetTagDWord("@UA_workstation_Assy2",assy2_count);
SetTagDWord("@UA_workstation_Assy3",assy3_count);
SetTagDWord("@UA_workstation_Assy4",assy4_count);


SetTagDWord("@UA_workstation_Job",6);  //6 indicates  value is written to table

uaArchiveUpdate(hArchive);                  //update the User Archive
uaArchiveClose(hArchive);                   //close the User Archive
uaDisconnect(hConnect);                 //close the Handle for the User Archive

return 0;
}
```

23. Next, compile the function by clicking the compile button  on the toolbar and save it as "station.pas"

24. Next, assign Trigger to this action. To do this click on the Info / Trigger icon  on the toolbar. This will open the dialog box shown below. Select the Trigger tab.

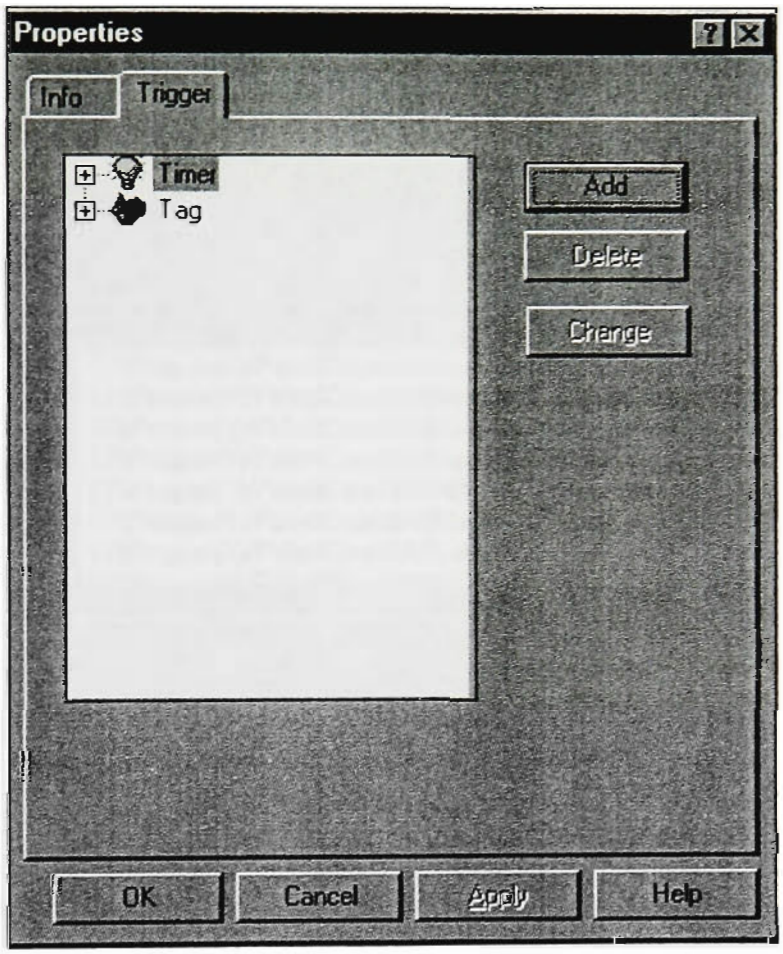


Fig C.21 Trigger dialog box

25. Select tag option and add a tag by clicking the Add button.

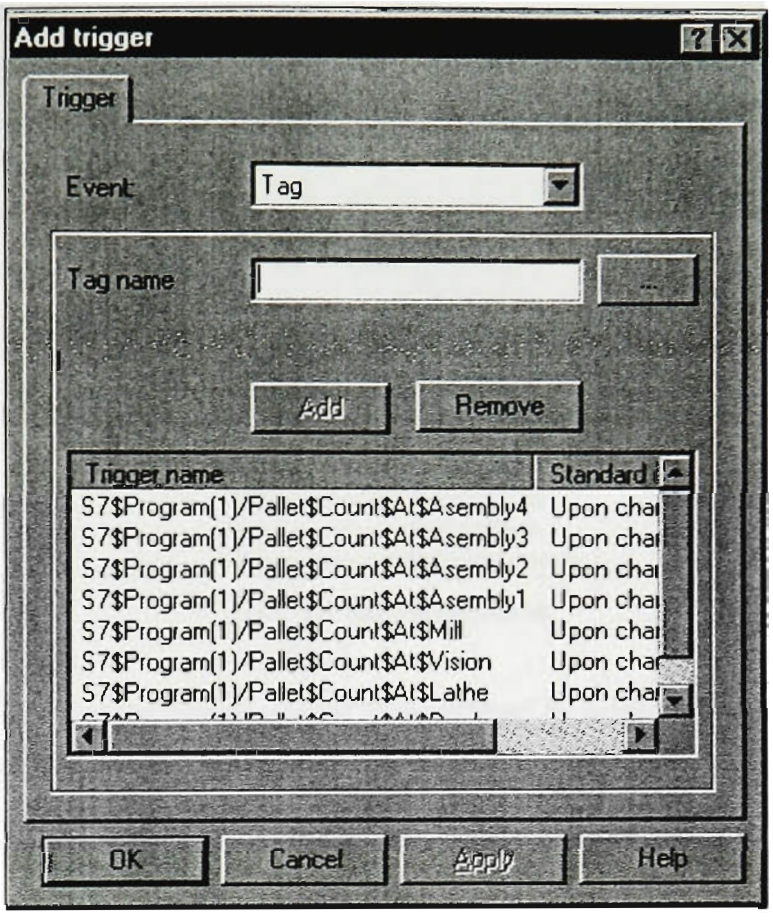


Fig C.22 Add Trigger dialog box

26. In the Add Trigger Button click the ellipse button for tag name. Expand down the Tag window to select S7$Program(1). This contains the Step 7 tags select the following tags:
    - S7$Program(1)/Pallet$CountAtVision
    - S7$Program(1)/Pallet$CountAtMill
    - S7$Program(1)/Pallet$CountAtLathe
    - S7$Program(1)/Pallet$CountAtDock
    - S7$Program(1)/Pallet$CountAtAsembly1
    - S7$Program(1)/Pallet$CountAtAsembly2
    - S7$Program(1)/Pallet$CountAtAsembly3
    - S7$Program(1)/Pallet$CountAtAsembly4

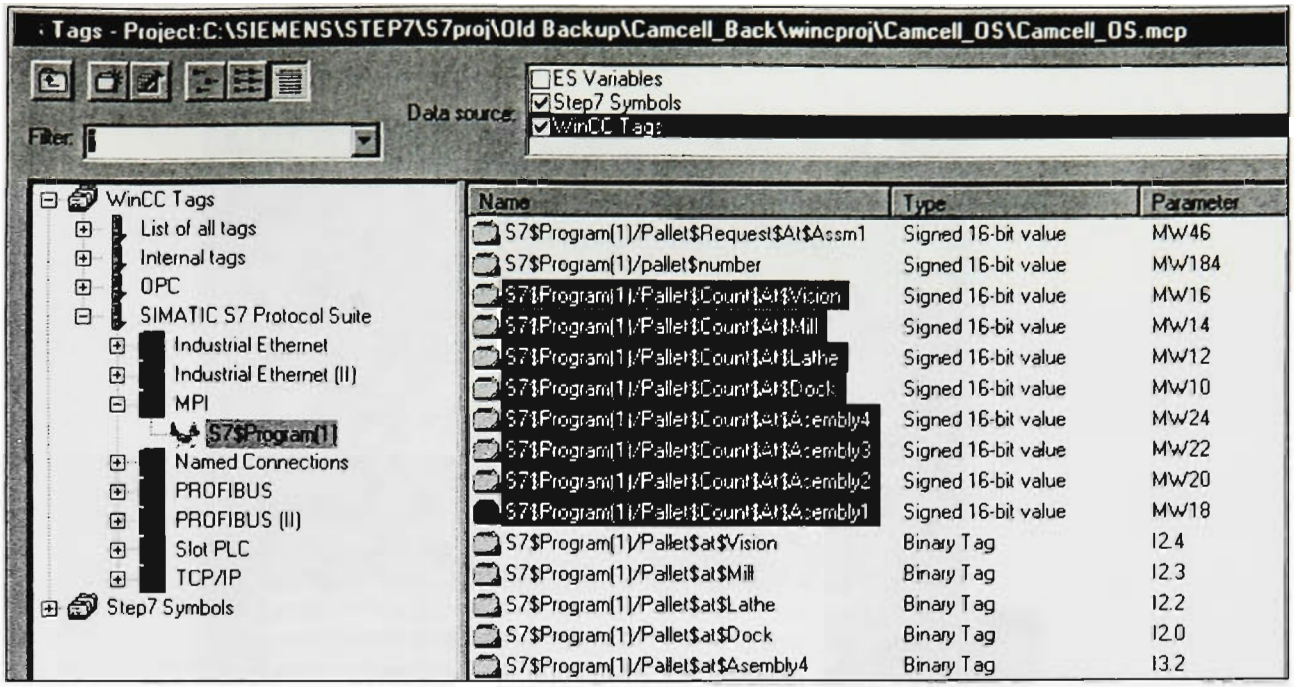


Fig C.23 Tag Management

27. Once the tags are added the "Standard cycle" is set to"2sec" by default. Right click on the standard cycle of each tag and change it to "Upon change" from "2sec". Click OK.

28. Recompile the script and save it.

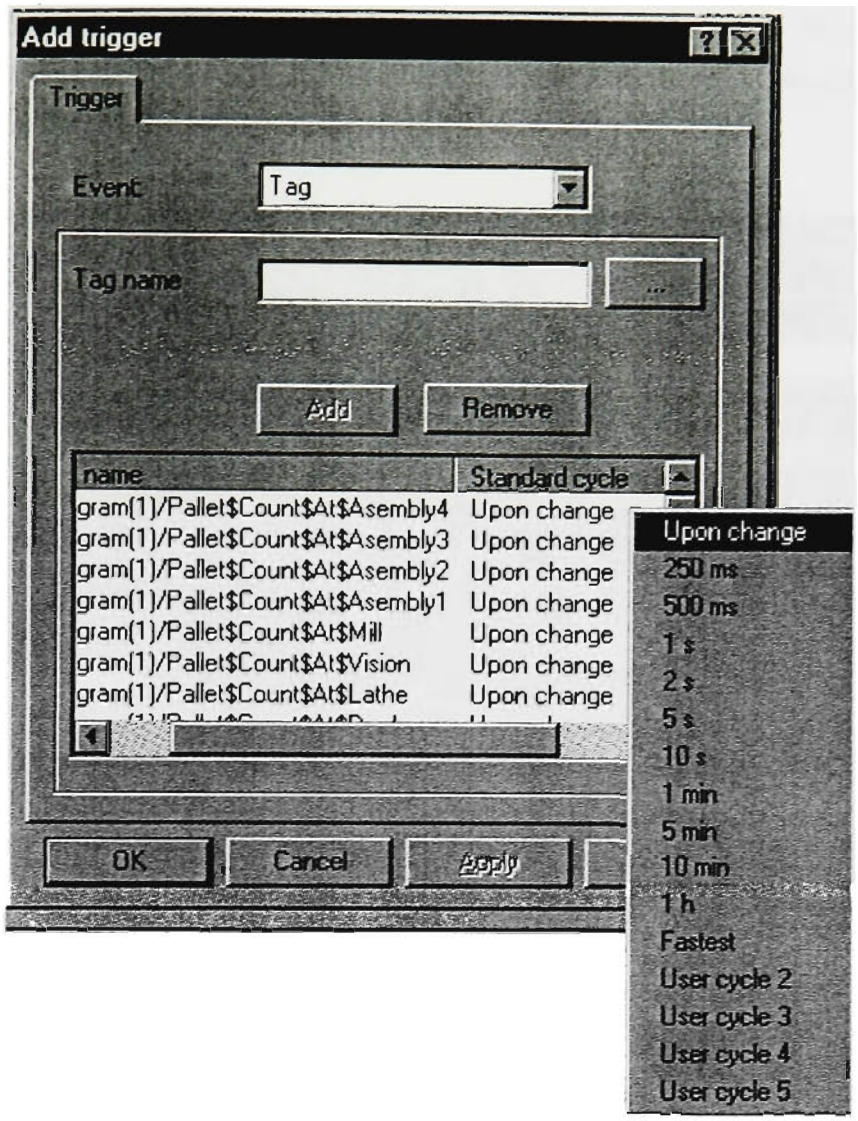


Fig C.24 Add trigger dialog box

29. Turn on the "Global Script Runtime" in the properties for the computer.

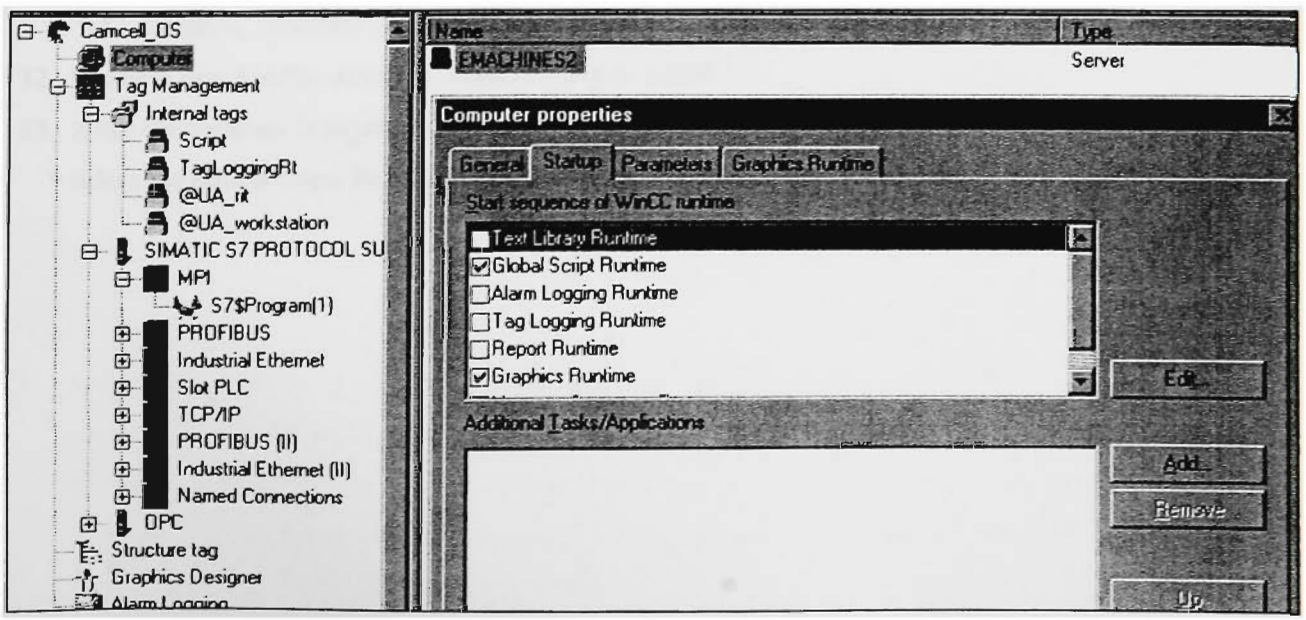


Fig C.25 Computer properties dialog box

30. Open a New Graphics Screen and insert a "WinCC User Archive-Table Element" control. Go to properties of the control and click on the "Select" button of the Source, select "workstation" from the dialog box. In the column tab select all the columns. We have selected the User Archive that was created; now we can view the data of all the station counters during run time mode of the project. Save the graphics as "camcell.pdl"

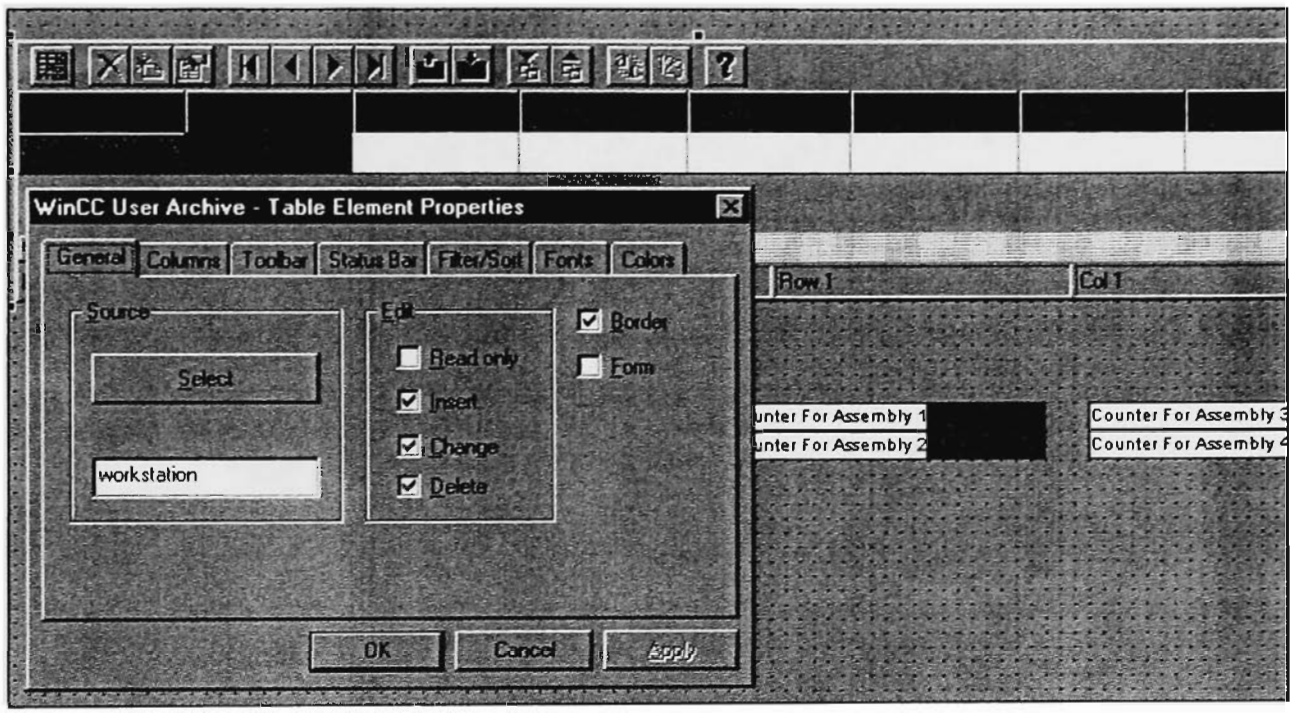


Fig C.26 User Archive table properties

31. So far we have completed the archiving procedure of the tags though Step 7 & WinCC. Now the last step is to create a Remote view in FoxPro so that the real time value of the pallet counter can be dumped the database. Before we create a Remote view we need to establish connection to WinCC Runtime database. WinCC uses "Sybase Adaptive Server Anywhere" to store all the Run time process values. For our project the name of the Sybase Runtime Database is "CC_Camcell__03-09-03_17:24:01R", this name is automatically assigned by Sybase.
32. Create a new FoxPro database. Name it "RealtimeDB"
33. After the database is saved, a database designer screen will open. Right click on the screen and select the option "New Remote View" Click on the button New View.

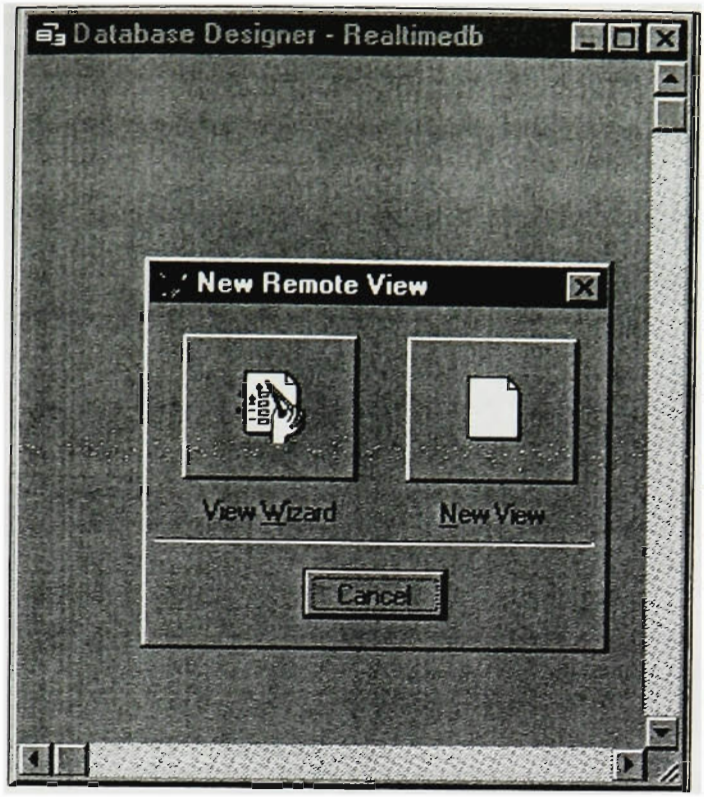


Fig C.27 New remote view

34. Click on New button in "Select Connection or Data Source"

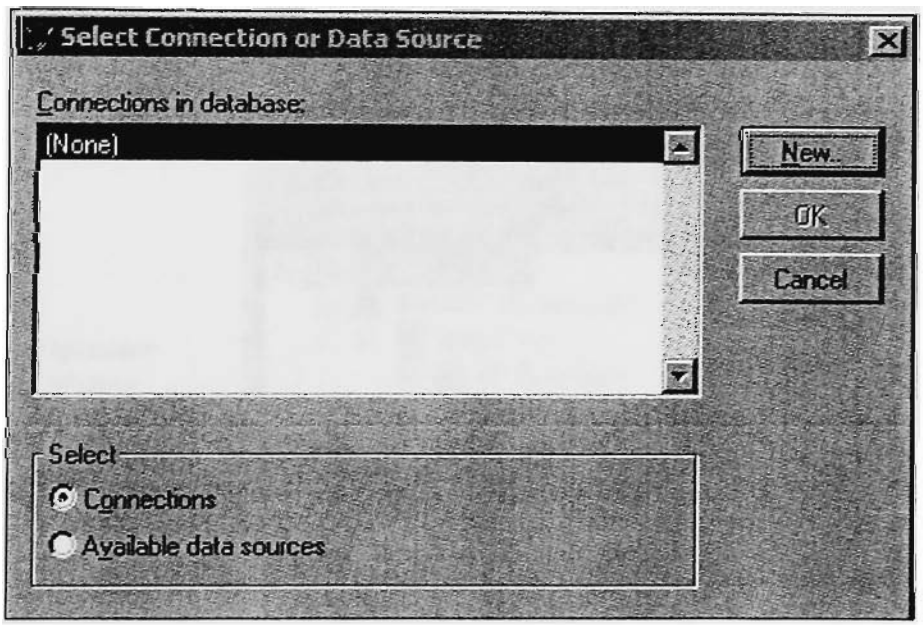


Fig C.28 Select Connection dialog box

35. A Connection Designer window shown below will open. We need to select appropriate "Data source" WinCC has created a Data Source Name (DSN) for us. If the list box on the "Data Source" is clicked the name in the following format appears **"CC_operator station name_ date/timerR".** In this application, the name is **"CC_Camcell__03-09-03_17:24:01R"**

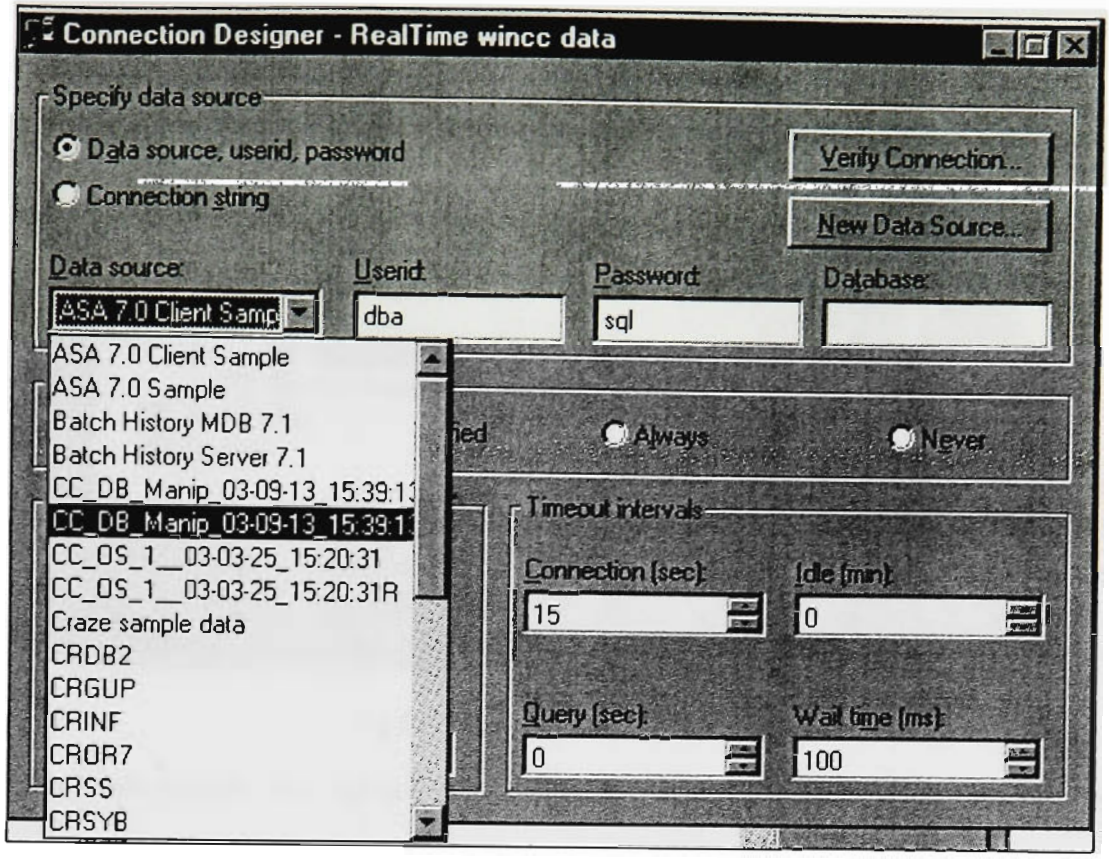


Fig C.29 Connection Designer

Note: Operator station name is in SIMATIC Manager. See the figure below.

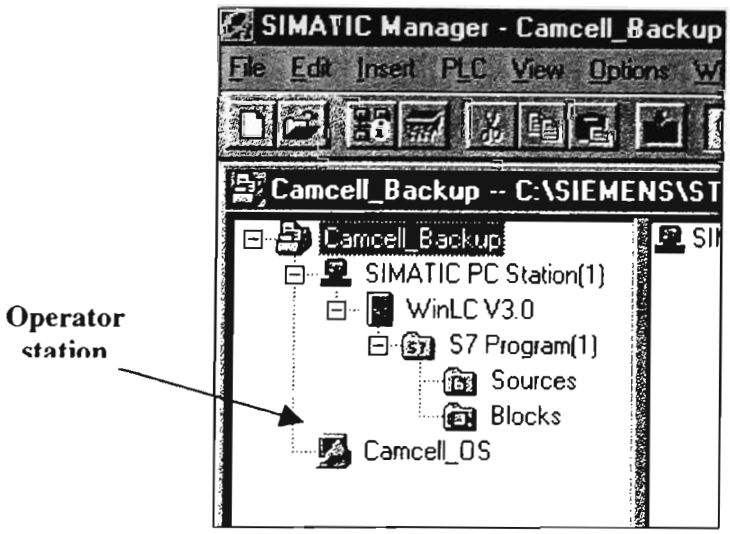


Fig C.30 Operation Station

36. In the Connection Designer put the Userid as **dba** and Password as **sql.** Check Display warnings option in Data processing. Now click on the button "Verify Option", you should get a messaging saying "Connection Succeed". Click OK.

37. Save the connection. In this application, the connection is names named as "RealTime wincc data". The connection between WinCC's Sybase database and FoxPro database is successfully established.

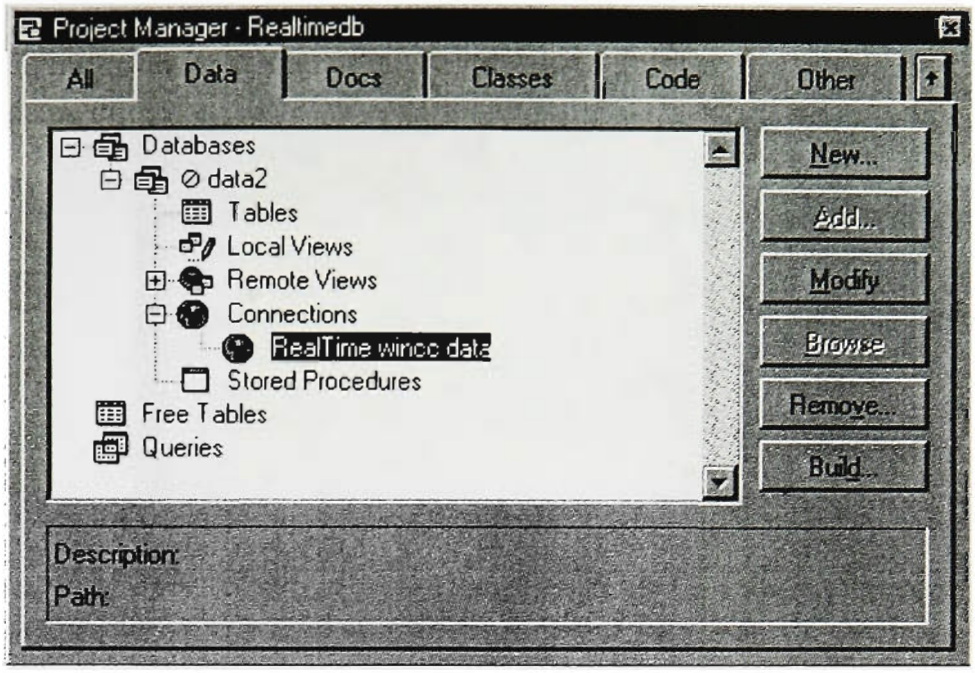


Fig C.31 FoxPro Connections

38. Create a new Remote View and select the "RealTime wincc data" from the connection. Click OK.

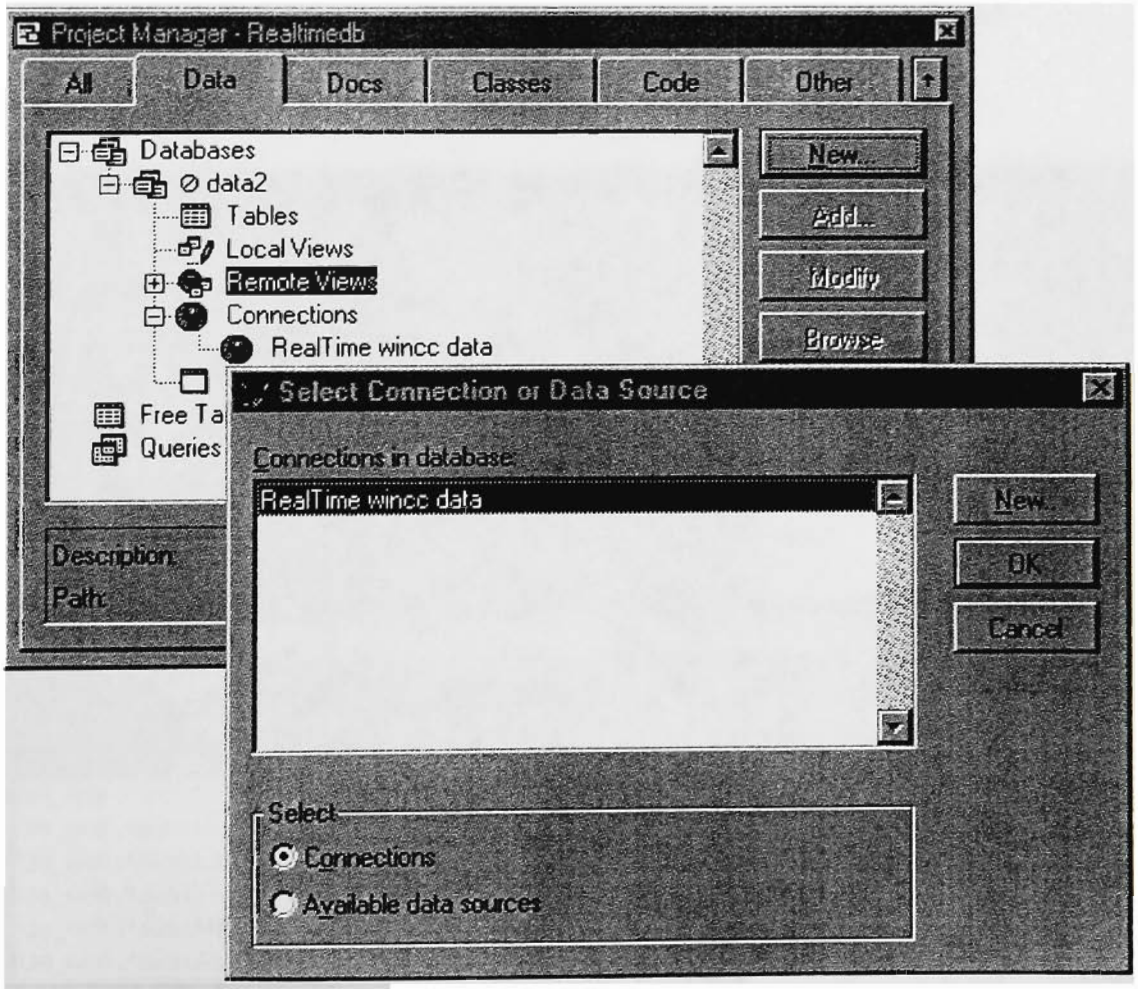


Fig C.32 Select Connections dialog box

39. From the selection box select "UA#workstation"

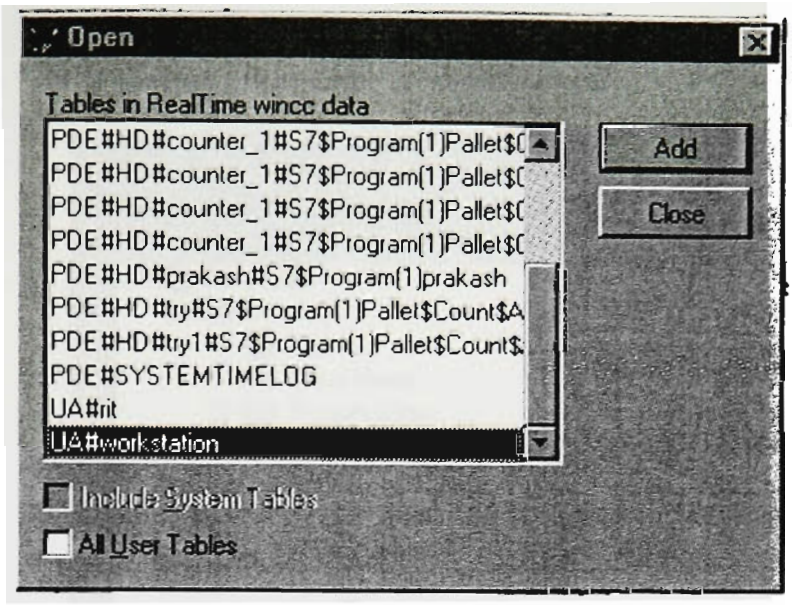


Fig C.32 Table selection

Note: One can browse through the Sybase database to get a feel of how the WinCC runtime database is configured. To open the Sybase enter "scview" in RUN from the start icon of the windows task bar. Click on connect database. USER ID: **dba** and PASSWORD: **sql**

40. Select all the fields from the table. Save the Remote View as "Workstation_count"

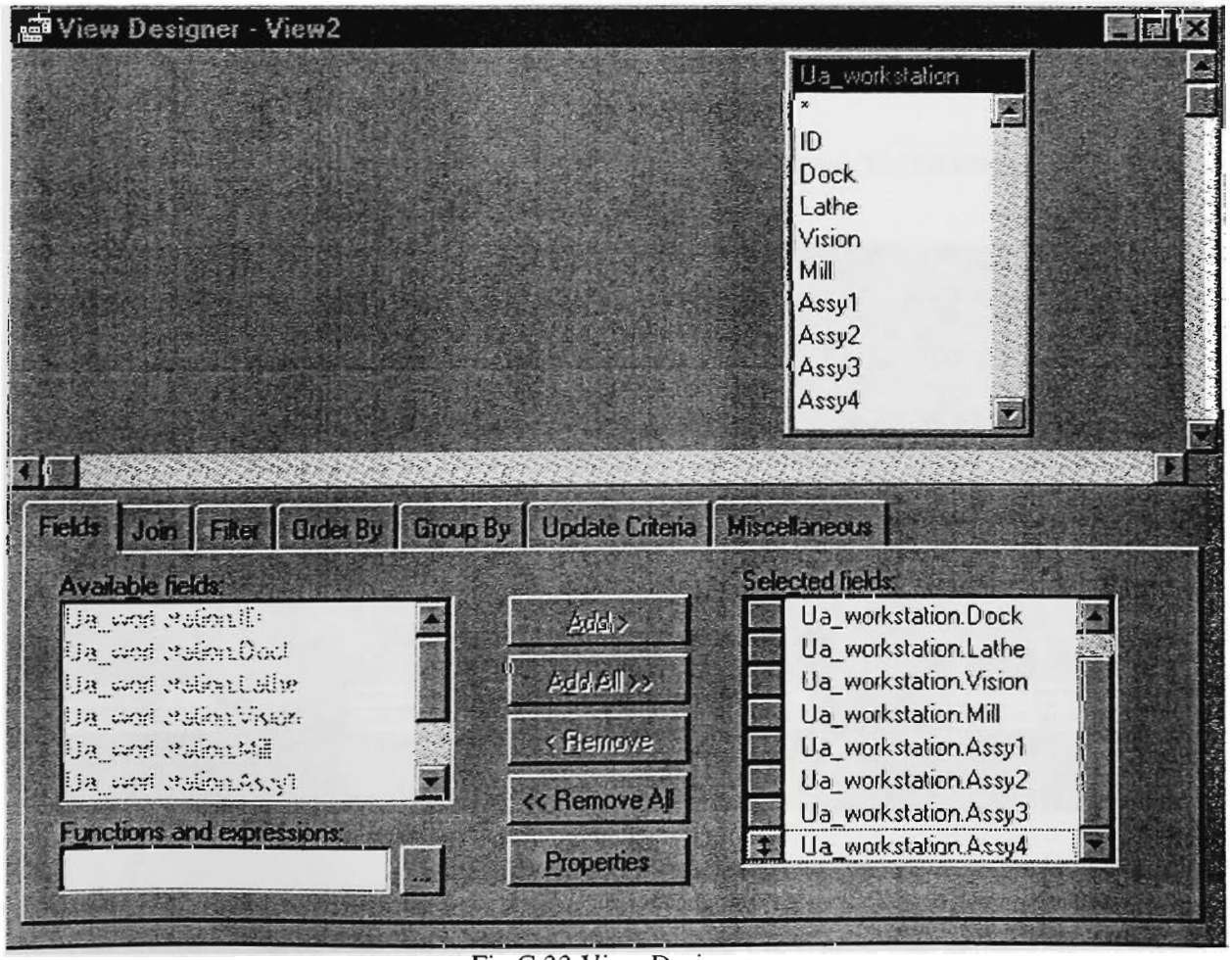


Fig C.33 View Designer

41. Your Remote view should look similar to the picture shown below.

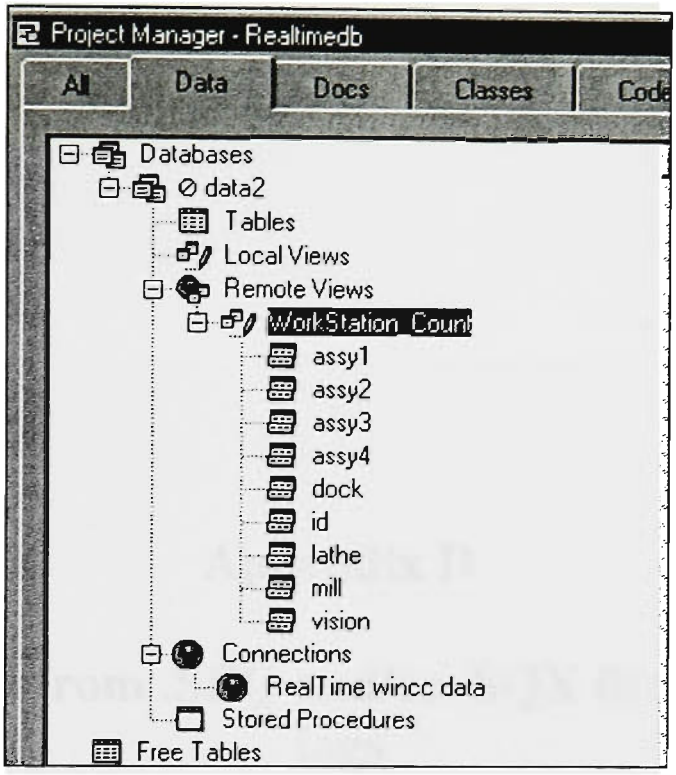


Fig C.34 Workstation_count Table

We have completed all the main steps involved in configuring a WinCC and FoxPro database for capturing the real-time data.

42. Keep the WinCC Graphics in RUN mode and start your application. See the changes in the table and check the data in FoxPro "Workstation_count" table.

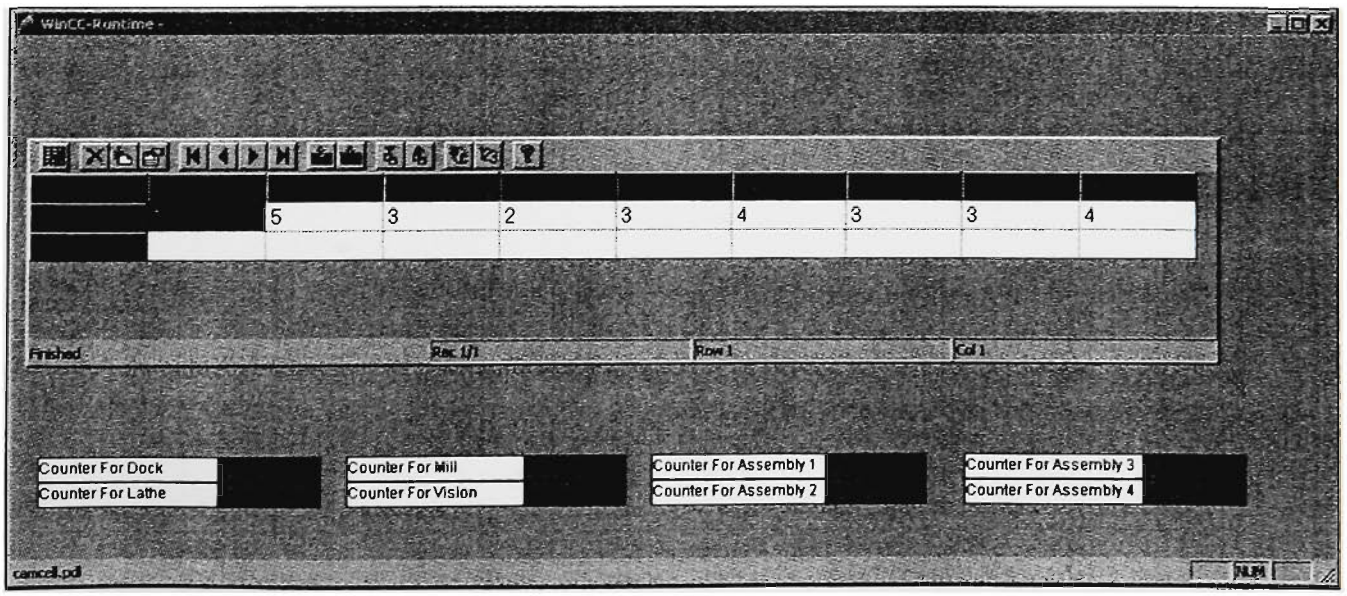


Fig C.35 WinCC Runtime

43. This procedure can be followed to capture Real-time Step 7 & WinCC data into any database by create appropriate ODBC connection.

Appendix D

Write Data from .SEQ and/or .SQX file to WinCC tags

Appendix D. Write Data from .SEQ and/or .SQX file to WinCC tags

Purpose
The purpose of this document is to give step-by-step instructions to write data from SEQ and/or .SQX to WinCC tags. This will involve reading SCREW.SEQ file residing in a known location.

Pre-requisite
SCREW.SEQ file should exist in C:\ drive of the computer. Refer Attachment 2 for .SEQ file format. Also WinCC tags must be created as per the list provided in Attachment 1 of appendix A.

Procedure:

1. Open the graphic screen and create a graphic as shown in Fig.D.1 below. Save the file as "Read_SEQ.pdl"

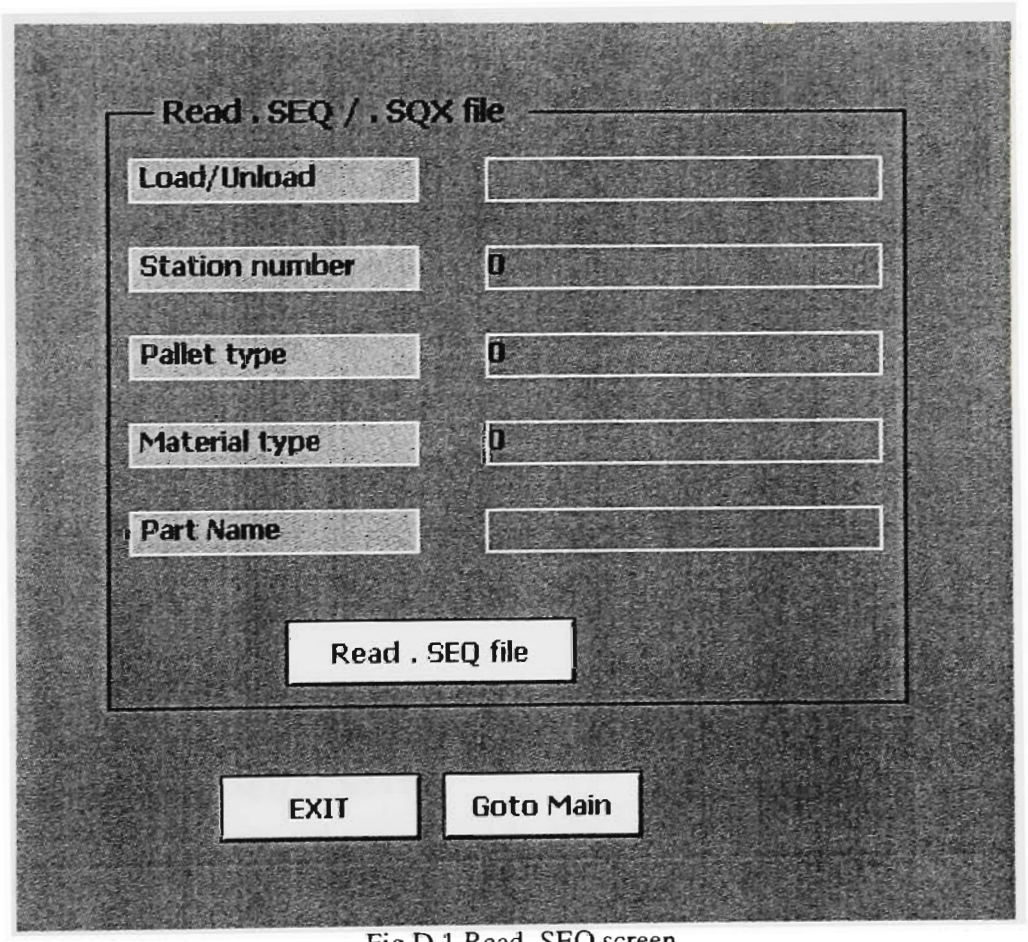


Fig.D.1 Read_SEQ screen

Configure the IOField as per the list below:

IOField	Tag	Update	Type
Load/Unload	Load_Unload	Upon change	Both
Station Number	Station_Number	Upon change	Both
Pallet Type	Pallet_Type	Upon change	Both
Material Type	Material_Type	Upon change	Both
Part Name	Part_Name	Upon change	Both

In order to configure the IOField, right click on the IOField and select the option "Configuration Dialog" from the pop up menu.

2. A modular approach is taken to open the file using a global function that can be used by any other function as well. This global file will be named as "infile()". The remaining script of reading the .SEQ files is done on "Read .SEQ file" button click event.

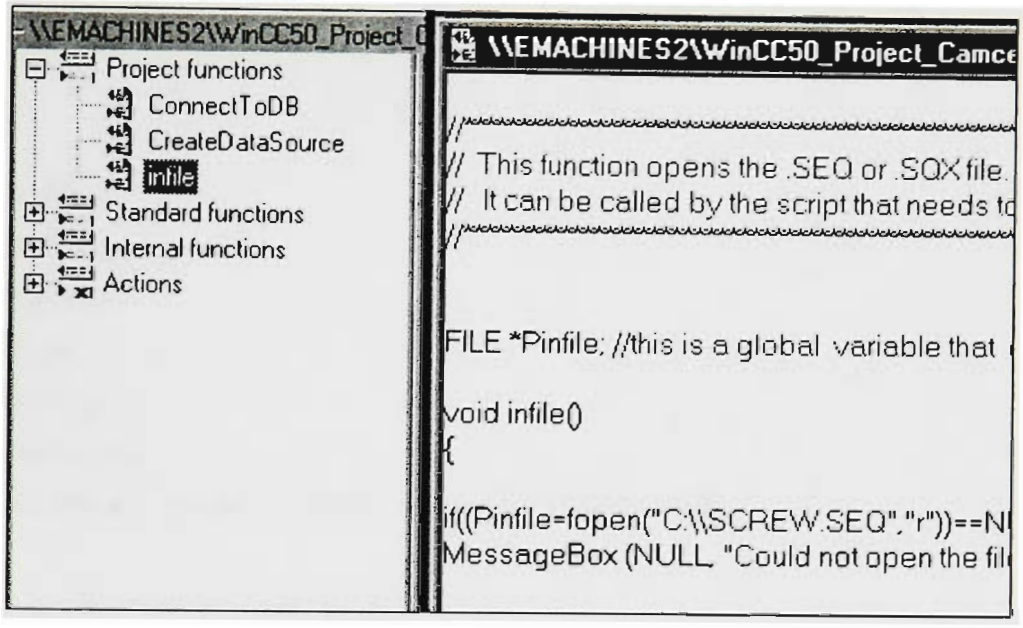


Fig D.2 Script for reading the .seq file

3. Create a new "Project function" Write the below script in the editor widow. After writing the script compile the function by clicking the compile button  on the toolbar. Save the file as "infile.fct"

```
//*************************************************************
// This function opens the .SEQ or .SQX file.
// It can be called by the script that needs to read the .SEQ or .SQX file
//*************************************************************

FILE *Pinfile; //this is a global variable that can be use in any function of this project

void infile()
{
if((Pinfile=fopen("C:\\SCREW.SEQ","r"))==NULL)      //open the .SEQ file
MessageBox (NULL, "Could not open the file", "ERROR", MB_OK | MB_ICONEXCLAMATION |
MB_SYSTEMMODAL);
}
```

4. Write the below script in the editor widow for "Read .SEQ file" button. This script read the open file and sets the respective tags. Once you are done writing the script compile the function by clicking the compile button  on the toolbar. Click OK to save.

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName,
UINT nFlags, int x, int y)
{

//declare variables
char LoadUnload[14];
int StationNumber;
int PalletType;
int MaterialType;
char PartName[14];
extern FILE *Pinfile; //pointer to the Pfile created in "infile" function

//Scan the file according to format specified.

if( fscanf(Pinfile,"%s %d %d %d %s",LoadUnload,&StationNumber,&PalletType,&MaterialType,Part
Name)==NULL)
  SetTagChar ("read_seq","Could not read the text");

    //set all the tags to respective variable
    SetTagChar ("Load_Unload",LoadUnload);
    SetTagDouble("Station_Number",StationNumber);    //Return-Type :BOOL
    SetTagDouble("Pallet_Type",PalletType);
    SetTagDouble("Material_Type",MaterialType);
    SetTagChar ("Part_Name",PartName);

}
```

5. Run the WinCC Graphics and keep clicking "Read .SEQ file" button to read the .SEQ file data.

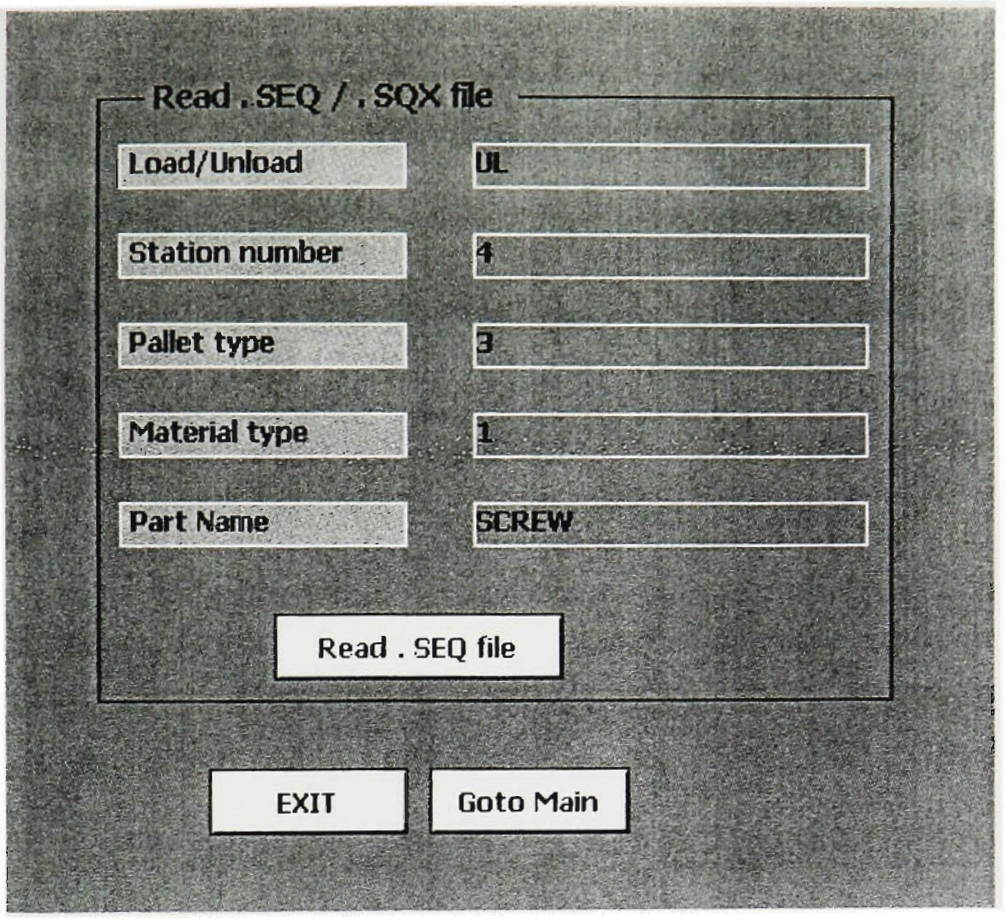


Fig D.3 Read .Seq screen

6. We have completed the procedure to read the data from SCREW.SEQ file and write it to WinCC tags.

Attachment 2

SCREW.SEQ file format

UL 4 3 1 SCREW
LD 1 3 1 SCREW
LD 1 3 5 SCREW
UL 1 3 6 SCREW
LD 2 3 6 SCREW
UL 2 3 7 SCREW
LD 4 3 7 SCREW

Appendix E

Handles for Runtime Archive Functions

Appendix E. Handles for Runtime Archive Functions

UaConnect	--> Handle UAHCONNECT	
	required by:	
	uaDisconnect	
	uaQueryArchive	--> Handle UAHARCHIVE
	uaQueryArchiveByName	--> Handle UAHARCHIVE
		required by:
		uaArchiveOpen
		Required for:
		uaArchiveClose
		uaArchiveDelete
		uaArchiveExport
		uaArchiveGetFieldLength
		uaArchiveGetFields
		uaArchiveGetFieldType
		uaArchiveGetFieldValueDate
		uaArchiveGetFieldValueDouble
		uaArchiveGetFieldValueLong
		uaArchiveGetFieldValueString
		uaArchiveGetFieldName
		uaArchiveGetFilter
		uaArchiveGetID
		uaArchiveGetName
		uaArchiveGetSort
		uaArchiveImport
		uaArchiveInsert
		uaArchiveMoveFirst
		uaArchiveMoveLast
		uaArchiveMoveNext
		uaArchiveMovePrevious
		uaArchiveReadTagValues
		uaArchiveReadTagValuesByName
		uaArchiveRequery
		uaArchiveSetFieldValueDate
		uaArchiveSetFieldValueDouble
		uaArchiveSetFieldValueLong
		uaArchiveSetFieldValueString
		uaArchiveSetFilter
		uaArchiveSetSort
		uaArchiveUpdate
		uaArchiveWriteTagValues
		uaArchiveWriteTagValuesByName
		UaReleaseArchive

Appendix F

Format of a .SQX file

Appendix F. Format of a .SQX file

LR 1 - request pallet with part on it

LM 2 - get raw material

LM 3 - load chuck

LF 0 - release (free) empty pallet

LD SCREW.LTH

LE - execute program

LT - tool change

LN 2 - notify ROUTER (that lathe is busy)

LR 1 - request pallet

LM 2 - get jig.

LC WAIT FOR MANIPULATOR TO FINISH

LF 0 - release (free) empty pallet

LT - tool change

LW 1 06 T - wait for C to go high

LM 4 - load jig on part

LM 8 - wait till complete

LP 0 11 - spindle start

LP 0 12 - pulse h input to cause a continue

LT - tool change

LT - tool change

LX - end of lathe program

LN 1 - NOTIFY ROUTER

LM 6 - unload part & jig.

LR 0 - request empty pallet

LM 7 - drop

LC WAIT FOR THE MANIPULATOR TO FINISH THE TASK

LM 1 - RETURN MANIPULATOR TO HOME POSITION

LF 1 - release (free) full pallet

LN 0 - notify ROUTER (that lathe is idle)